



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



Rodrigo Humberto Fuentes Rodríguez

Sistemas de Aprendizaje Automático con Cloud Computing

Informe de Proyecto de Título de Ingeniero Civil Electrónico



**Escuela de Ingeniería Eléctrica
Facultad de Ingeniería**

Valparaíso, 17 de abril de 2018



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO

Sistemas de Aprendizaje Automático con Cloud Computing

Rodrigo Humberto Fuentes Rodríguez

Informe Final para optar al título de Ingeniero Civil Electrónico,
aprobada por la comisión de la
Escuela de Ingeniería Eléctrica de la
Facultad de Ingeniería de la
Pontificia Universidad Católica de Valparaíso

Conformada por:

Sr. Gonzalo Farías Castro

Profesor Guía

Sr. Gabriel Hermosilla Vigneau

Segundo Revisor

Sr. Sebastián Fingerhuth Massmann

Secretario Académico

Valparaíso, 17 de Abril de 2018

Resumen

El presente informe, trata sobre el estudio y uso de entornos Cloud Computing (Computación en la nube) para la creación de aplicaciones que requieran de Machine Learning (Aprendizaje Automático). Para ello, se utilizaron distintos proveedores destacados en la industria tales como: Microsoft (Microsoft Azure Studio), Amazon (Amazon Machine Learning) y Google (Google Cloud Platform).

Se comparan los servicios entregados por las compañías ya mencionadas con el fin de realizar futuros proyectos en alguna de estas plataformas, para lograrlo, se crean dos aplicaciones (las mismas para cada proveedor) que resuelven un problema de reconocimiento de patrones, para luego evaluar su desempeño. La primera aplicación consiste en un clasificador de 2 clases, capaz de etiquetar datos nuevos en un eje cartesiano basándose en una base de datos previa, mientras que la segunda aplicación funciona como un clasificador de imágenes para un proyecto de fusión nuclear, en donde la base de datos inicial se debe procesar de cierta manera (reconstruir la información para luego reducirla de tal manera que sea lo más fiel posible de la información inicial) para poder ser usado como datos de entrada para el entrenamiento y evaluación del sistema. Finalmente, se comparan las plataformas de Cloud Computing estudiadas de manera cuantitativa (con los modelos de Machine Learning utilizados en los experimentos, destacando aquel que entrega los mejores resultados) y cualitativa (mencionando características propias de cada plataforma y como también ventajas y desventajas).

Palabras claves: Machine Learning, Cloud Computing, Big Data, Microsoft Azure Studio, Python, Amazon Web Services (AWS), Amazon Machine Learning (AML), Google Cloud Platform y Tensorflow.

Abstract

The following report is about the study and use of Cloud Computing environments, for the creation of applications that require Machine Learning. For this, different leading providers in the industry were used, such as: Microsoft (Microsoft Azure Studio), Amazon (Amazon Machine Learning) and Google (Google Cloud Platform).

The services delivered by the aforementioned companies are compared in order to carry out future projects in any of these platforms, to achieve this, two applications are created (the same for each provider) that solves a problem of pattern recognition, to then evaluate their performance. The first application consists of a classifier of 2 classes, capable of labeling new data in a cartesian axis based on a previous database, while the second application works as an image classifier for a nuclear fusion project, where the base Initial data should be processed in a certain way (reconstruct the information to then reduce it in such a way that it is as faithful as possible of the initial information) to be used as input data for the training and evaluation of the system. Finally, we compare the Cloud Computing platforms studied in a quantitative way (with the Machine Learning models used in the experiments, highlighting the one that delivers the best results) and qualitative (mentioning the characteristics of each platform and also advantages and disadvantages).

Keywords: Machine Learning, Cloud Computing, Big Data, Microsoft Azure Studio, Python, Amazon Web Services (AWS), Amazon Machine Learning (AML), Google Cloud Platform and Tensorflow.

Índice general

Introducción.....	1
Objetivos Generales	2
Objetivos Específicos	2
1 Antecedentes Generales y Propuesta	3
1.1 Descripción del Problema	3
1.2 Descripción de la Propuesta	3
1.3 Cloud Computing (Computación en la nube).....	3
1.3.1 Servicios de Cloud Computing.....	3
1.3.2 Tipos de nube	4
1.3.3 Ventajas y Desventajas.....	4
1.4 Big Data.....	5
1.5 Machine Learning.....	5
1.5.1 Objetivo del Machine Learning.....	5
1.5.2 Aplicaciones de Machine Learning.....	6
1.5.3 Tipos de aprendizaje	6
1.5.4 Modelos de Machine Learning.....	6
1.6 Minería de Datos.....	11
1.6.1 Aplicaciones de la Minería de Datos.....	12
1.7 Trabajo a realizar	13
1.7.1 Etiquetador de Datos (Ejemplo ilustrativo)	13
1.7.2 Clasificador de imágenes Thomson.....	14
2 Microsoft Azure Studio.....	19
2.1 Microsoft Azure Studio (Cloud Computing).....	19
2.2 Etiquetador de Datos en Microsoft Azure Studio	20
2.3 Clasificador de Imágenes Thomson en Microsoft Azure Studio	22
3 Amazon Web Services (AWS)	27
3.1 Amazon Web Services (Cloud Computing)	27
3.1.1 Almacenamiento S3	27
3.1.2 Amazon Machine Learning	28
3.2 Etiquetador de Datos en Amazon Web Services	31
3.3 Clasificador de Imágenes Thomson en Amazon Web Services	32
4 Google Cloud Platform (GCP).....	35
4.1 Google Cloud Platform (Cloud Computing).....	35
4.1.1 Almacenamiento en “Storage”	35
4.1.2 Compute Engine.....	36
4.2 Etiquetador de Datos en Google Cloud Platform.....	38
4.3 Clasificador de Imágenes Thomson en Google Cloud Platform.....	39

5 Comparación de las plataformas	42
5.1 Comparación cuantitativa de los resultados obtenidos en el clasificador de imágenes Thomson.....	42
5.1.1 Datasize de 15x16 pixeles.....	42
5.1.2 Datasize de 8x7 pixeles.....	43
5.2 Comparación cualitativa entre las plataformas	44
Discusión y Conclusiones	46
Bibliografía	48

Introducción

Desde siempre, la información ha sido un valioso recurso que ha ayudado al hombre de múltiples formas, ya sea para cambiar su manera de pensar, de actuar, tomar decisiones, etc. Debido a esto, se ha querido registrar todo tipo de datos, con el fin de que sea de utilidad en algún futuro, trayendo como consecuencia una tasa de generación de éstos demasiado grande, al punto de ser incapaz de procesarlos, perdiendo así su objetivo. Es por eso que se crearon técnicas informáticas que ayudasen al análisis de esta información, dando origen al Machine Learning.

El Machine Learning o Aprendizaje Automático es un subcampo de la computación e Inteligencia Artificial, se basa en otorgarle a las máquinas la capacidad de aprender sobre algún tema tal como lo haría un humano, es decir, entregándole de antemano información que estudie con la que formará algunos criterios, para que luego ser capaz de aplicar sus conocimientos en aplicaciones tales como: predecir valores futuros de alguna secuencia numérica, reconocer y clasificar imágenes, detectar patrones en datos, etc. Tal como lo haría un humano, con la diferencia de no cometer errores por la fatiga de realizar trabajos repetitivos y con la rapidez que solo un computador puede ofrecer (la cual es superior a la humana).

El uso del Machine Learning saca su provecho cuando se aplica con grandes cantidades de información, es por ello que se convierte en una excelente alternativa a la hora de procesar Big Data. Los principales usuarios de este tipo de tecnologías son las empresas, las cuales acumulan grandes cantidades de información que necesitan para poder llevar a cabo sus negocios, pero para poder llevar a la práctica el análisis de Big Data con Machine Learning se requiere de computadores con buenas características (disco duro, memoria RAM, etc.) debido a que se requiere los resultados lo antes posible (o al menos que se demore un tiempo prudente), es por ello que se tiene que invertir constantemente en gastos computacionales tales como renovación de equipos, servicios técnicos, lugar físico de los computadores, etc.

A modo de alternativa para cubrir las necesidades ya mencionadas de la computación local (entiéndase por los recursos que maneja el usuario) es que se desarrolla la tecnología del Cloud Computing o Computación en la Nube, la cual que permite ofrecer servicios computacionales tales como procesamiento de datos, almacenamiento y acceso a la información (entre otros), sin

la necesidad de utilizar recursos de manera local. Es por ello que surgen empresas que ofrecen este tipo de servicios, las cuales serán estudiadas a continuación.

Objetivos Generales

- Evaluar e implementar un sistema de reconocimiento de patrones y aprendizaje automático utilizando entornos de Cloud Computing.

Objetivos Específicos

- Estudiar los entornos de Cloud Computing disponibles para construir sistemas de reconocimiento de patrones.
- Seleccionar un entorno de Cloud Computing y un problema de Reconocimiento de Patrones
- Diseñar e implementar el problema seleccionado utilizando las herramientas del entorno virtual.

1 Antecedentes Generales y Propuesta

1.1 Descripción del Problema

Se premisa consiste en la necesidad de poder procesar cantidades importante de información (en el orden de los terabits) dentro de un rango prudente de tiempo (menor tiempo posible), para poder llevar a cabo esta labor, se requiere de una buena infraestructura computacional además de técnicas avanzadas de programación con las cuales realizar un análisis inteligente de los datos.

1.2 Descripción de la Propuesta

La solución propuesta consiste en estudiar y utilizar servicios de computación en la nube (Cloud Computing) de diferentes proveedores (Microsoft Azure Studio, Amazon Web Services y Google Cloud Platform) para así evitar las restricciones computacionales (tanto a nivel de software como de hardware) que se puedan llegar a presentar, además estas plataformas permiten el uso de Machine Learning, lo cual se aplica con el fin de reducir considerablemente los tiempos de procesamiento de la información.

1.3 Cloud Computing (Computación en la nube)

Es una tecnología que ofrece servicios de computación (almacenamiento de archivos, software, procesamiento de datos, etc.) por medio de internet, sin la necesidad de instalar y/o trabajar con recursos locales (entiéndase trabajo local como los recursos computacionales que dispone el usuario). Su objetivo principal es que el usuario no dependa de una ubicación física para trabajar, además de que pueda acceder a múltiples servicios dentro de la misma nube [1].

1.3.1 Servicios de Cloud Computing

Los servicios de Cloud Computing son variados y se pueden clasificar en 3 tipos [1] [2]:

- **Software como servicio (SaaS):** Este servicio se encuentra en la capa más alta, le brinda al usuario una infraestructura (computadores, almacenamiento, etc.), correcto funcionamiento de las aplicaciones y mantenimiento tanto de la infraestructura como la de sus aplicaciones (nuevas versiones, corrección de bugs, etc.).

- **Plataforma como servicio (PaaS):** Este servicio es la capa del medio, le brinda al usuario una plataforma de desarrollo y las herramientas de programación para poder crear sus propias aplicaciones, sin controlar la infraestructura.

- **Infraestructura como Servicio (IaaS):** Este servicio se encuentra en la capa inferior, le brinda al usuario un almacenamiento básico, servidores, enrutadores y otros sistemas que le permitan manejar tipos específicos de cargas de trabajo.



Figura 1-1: Capas del servicio de Cloud Computing [3]

1.3.2 Tipos de nube

En Cloud Computing, se pueden distinguir 3 tipos de nubes, las cuales son [2]:

- **Nube Pública:** Son mantenidas y gestionadas por terceras personas, se suministra del servicio a miles de usuarios de manera simultánea e independiente.

- **Nubes Privadas:** El usuario se abastece así mismo con los servicios de la nube, de forma que solo él tiene acceso.

- **Nubes Híbridas:** Es una mezcla entre las nubes pública y privada, donde el cliente es propietario de cierta parte de la nube, que a su vez comparte con otros usuarios.

1.3.3 Ventajas y Desventajas

A continuación, en la Tabla 1-1 se presentan algunas de las principales ventajas y desventajas que presentan los servicios de Cloud Computing [4].

Tabla 1-1: Ventajas y Desventajas de Cloud Computing.

Ventajas	Desventajas
El usuario no necesita preocuparse por las limitaciones de software o hardware.	Requiere de una conexión a internet para poder acceder al servicio.
Se puede acceder al servicio desde cualquier ubicación, por lo que usuarios de distintas regiones pueden trabajar con los mismos registros.	Se corre el riesgo de que algún delincuente informático pueda tener acceso a los datos, ya que la data de cada cliente debe recorrer diferentes nodos antes de llegar a su destino.
Entrega prestaciones de servicios a nivel mundial, lo que reduce los tiempos de inactividad al mínimo para el proveedor.	Centralización de almacenamiento de datos y aplicaciones, lo que genera una dependencia del usuario hacia el proveedor.
Se cuenta con varias copias de seguridad, por lo que es más fácil recuperar datos.	La confiabilidad del servicio depende del estado tecnológico del proveedor.
El cliente no se necesita invertir en infraestructura o mantenciones de equipo.	Las aplicaciones están constantemente cambiando sus interfaces.

1.4 Big Data

Es un conjunto extremadamente grande de datos, aunque este concepto no se refiere a una cantidad específica, es posible hablar de Big Data al tratarse de archivos que están desde el orden de los gigabits en adelante, de manera tal que los algoritmos y software convencionales no podrían procesar y administrar toda esa información en un tiempo prudente. Existen tres tipos de Big Data: datos estructurados (son de longitud y formato definidos), datos no estructurados (carecen de algún formato específico) y datos semiestructurados (sus datos no se limitan a campos determinados) [5].

1.5 Machine Learning

Es un subcampo de las ciencias computacionales y una rama de la inteligencia artificial, se basa en que las máquinas aprendan, para que puedan tomar decisiones por sí mismos, esto se consigue teniendo una base de datos para entrenar al sistema y algoritmos que permitan generalizar comportamientos con el fin de poder predecir comportamientos a futuros a partir de lo ocurrido en el pasado/presente [6].

1.5.1 Objetivo del Machine Learning

Su objetivo es poder crear sistemas capaces de aprender por si mismos a partir de un conjunto de datos sin la necesidad de ser programados de manera explícita, gracias a eso, es posible procesar

una cantidad enorme de información (Big Data) para así actuar de manera eficaz y eficiente frente a problemas que son complejos y/o engorrosos para el ser humano.

1.5.2 Aplicaciones de Machine Learning

A pesar de que el ser humano posee niveles de percepción capaces de captar la información necesaria, procesarla, generar criterios gracias a que posee tipos de inteligencia (lingüística, musical, interpersonal, emocional, etc.) de las cuales la máquina carece, aún no es capaz de hacer todo ese trabajo con la velocidad que puede hacerlo una computadora, es por eso que se entrena para que parcialmente aprenda y “piense” como una persona para una tarea en específico. Puede aplicarse en diversas áreas tales como: diagnósticos médicos, motores de búsqueda, detección de fraude en el uso de tarjetas de crédito, clasificación en secuencias de ADN, reconocimiento de patrones, etc., en general se puede aplicar en cualquier caso que se quiera distinguir o clasificar elementos según alguna característica bien definida.

1.5.3 Tipos de aprendizaje

Se entiende como aprendizaje al proceso de adquirir nuevas habilidades o conocimiento a partir de la experiencia, estudio, razonamiento y observación, aunque para la máquina esto se limita a lo que puede adquirir mediante la experiencia. Se pueden distinguir distintos tipos de aprendizaje, tales como [7]:

- **Aprendizaje Supervisado:** El modelo se lleva a cabo con una base de datos previamente etiquetados, por lo que el sistema conoce de antemano la clasificación correcta (este tipo de aprendizaje es el que será aplicado en clasificador de imágenes del proyecto de fusión nuclear Thomson).
- **Aprendizaje no Supervisado:** El modelo se lleva a cabo solamente con los datos de entrada, y sin conocer la clasificación correcta (este tipo de aprendizaje es el que será aplicado en la aplicación etiquetadora de clases).
- **Aprendizaje Semi-supervisado:** El modelo presenta información mixta, es decir, tiene datos etiquetados y no etiquetados.
- **Aprendizaje por Refuerzo:** El algoritmo aprende mediante un flujo bidireccional de información (de la máquina al mundo y del mundo a la máquina), en base al ensayo-error.
- **Aprendizaje Multi-tarea:** Engloba métodos de aprendizaje que usan conocimiento aprendido para solucionar problemas parecidos a los ya vistos.

1.5.4 Modelos de Machine Learning

Se sabe que el fin de Machine Learning es crear sistemas que tengan la capacidad de aprender por ellos mismos, pero para lograrlo, necesitan de un modelo o algoritmo que les diga cómo actuar frente a determinados casos, es decir, el modelo es la “forma de pensar” que ocupa la máquina

para poder tomar decisiones. A continuación, se presentan algunos modelos de Machine Learning, los cuales fueron utilizados durante el transcurso de este proyecto.

- **Árbol de decisiones:**

Es un método analítico que usa una representación esquemática de las alternativas posibles que se pudiesen generar a partir de un determinado evento, los elementos de un árbol de decisiones son: los nodos (que representan a las características) y ramas (que representan a las reglas de clasificación a una etiqueta específica). La construcción de un árbol de decisión es la siguiente: se comienza con un nodo inicial denominado raíz, el cual tiene asociado una característica y una condición de 2 alternativas, cada una de esas alternativas llega a otro nodo por medio de una rama hasta que finalmente, la combinación de características que se formó en el árbol es suficiente para formar un criterio de clasificación. A continuación en la Figura 1-2, se presenta gráficamente como es un árbol de decisiones.

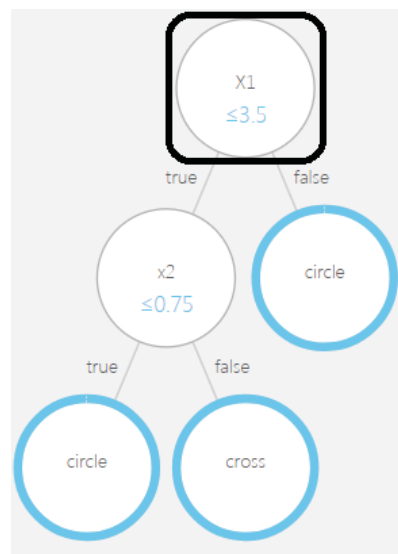


Figura 1-2: Esquema del modelo Árbol de Decisiones (creado en Microsoft Azure Studio).

En la Figura 1-2, se observa que el nodo encerrado en un cuadrado (el de más arriba) es la raíz, que de ella salen ramas que conectan a otros nodos y los nodos finales son aquellos que cumplen con todas las condiciones desde la raíz hasta su destino, sus círculos están remarcados y dentro contienen una etiqueta de la característica objetivo indicando a que clase pertenecen los datos que cumplen dichos requisitos [8] [9].

- **Bosque de Decisiones:**

Es un modelo de Machine Learning para la clasificación para variables objetivo que sean multiclase, está constituido por un conjunto de árboles de decisión, de los cuales, cada uno de

ellos actúa y clasifica de manera independiente según sus propios criterios. Para cada árbol, se genera un histograma con las etiquetas, que luego se suman para posteriormente normalizarse y obtener una probabilidad. La decisión final de este modelo es una votación ponderada entre las diferentes respuestas que se pueden llegar a generar, dándole mayor importancia a aquellos árboles que tengan mayor confianza en la predicción que presentan. A continuación en la Figura 1-3, se presenta la estructura grafica de un bosque de decisiones [9].

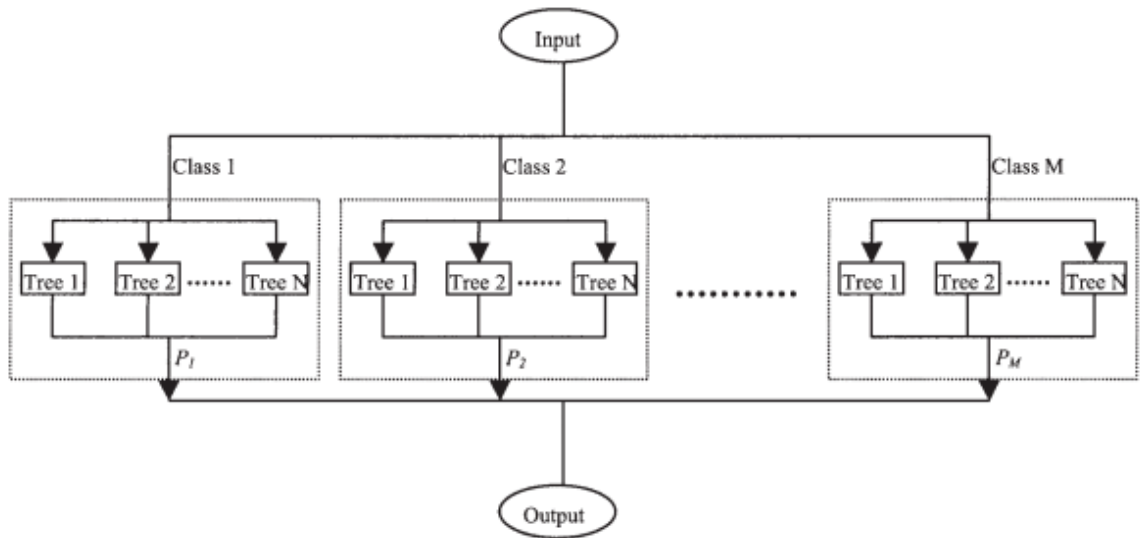


Figura 1-3: Esquema del modelo Bosque de Decisiones [10].

- **Jungla de Decisiones:**

Es un modelo de Machine Learning y extensión del Bosque de decisiones, es usado para resolver problemas de clasificación en donde la variable objetivo es multiclase, se construye de manera similar a un Bosque de Decisiones con la diferencia que sus nodos no se repiten, ya que sus árboles interactúan entre sí generando un solo gran esquema. Sus principales ventajas (con respecto al bosque de decisiones) son que minimizan los efectos de las características ruidosas y permiten fusionar ramas de los árboles de decisión con el fin de reducir los requisitos de memoria. A continuación en la Figura 1-4, se presenta un esquema del modelo Jungla de Decisiones [9].

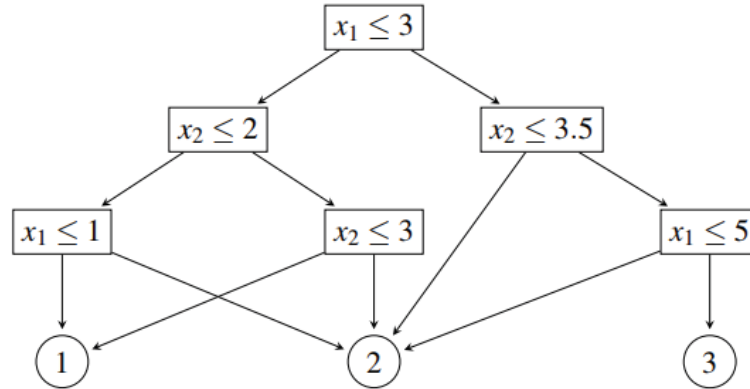


Figura 1-4: Esquema del modelo Jungla de Decisiones [11].

- **Uno vs Todos:**

Es un modelo de Machine Learning que le permite a algoritmos de clasificación de 2 clases, convertirse en algoritmos de clasificación multiclase. En esencia, lo que hace es crear un conjunto de modelos individuales y los resultados se combinan para crear un modelo que preside todas las clases, es por eso que cualquier clasificador de dos clases se puede utilizar como base para un modelo de “Uno vs Todos”.

- **Regresión Logística Multinomial:**

Es un modelo de predicción de probabilidad de ocurrencia de una variable categórica, que transforma términos de origen cualitativo y les asigna valores cuantitativos. El cambio de naturaleza de la variable consiste en crear y asignar variables auxiliares numéricas, para poder realizar cálculos matemáticos, con el fin de poder predecir en base a información cualitativa. Para la formulación del modelo, se necesita de una transformación logarítmica (logit) para cada categoría de la variable objetivo, tal como se muestra en la ecuación 1-1 [12].

$$\ln \left[\frac{p(X)}{1 - p(X)} \right] = b_o + \sum_{s=1}^n b_s x_s \quad (1-1)$$

De la ecuación 1-1 se tiene lo siguiente:

$p(X)$: Probabilidad que se tenga la característica en el conjunto de las covariables.

X : Conjunto de “n” covariables $\{x_1, x_2, \dots, x_n\}$.

b_o : Constante del modelo o término independiente.

b_n : Coeficientes de las covariables.

n : Cantidad de categorías de la variable dependiente menos una.

- Redes Neuronales:

Es un modelo Machine Learning que funciona similar a lo que haría un cerebro humano, en donde las neuronas son nodos que están organizadas por niveles o capas, cada nodo está conectado a un determinado conjunto de nodos del siguiente nivel, existen 3 tipos de nodos en una Red Neuronal los cuales son: nodos de entrada (contienen información de la base de datos), capa oculta (se les denomina a todas aquellas capas intermedias entre la entrada y salida, las cuales interactúan entre sí para llegar a un resultado) y la capa de salida (es el conjunto de características que se reconoce para la clasificación), tal como se observa en la Figura 1-5.

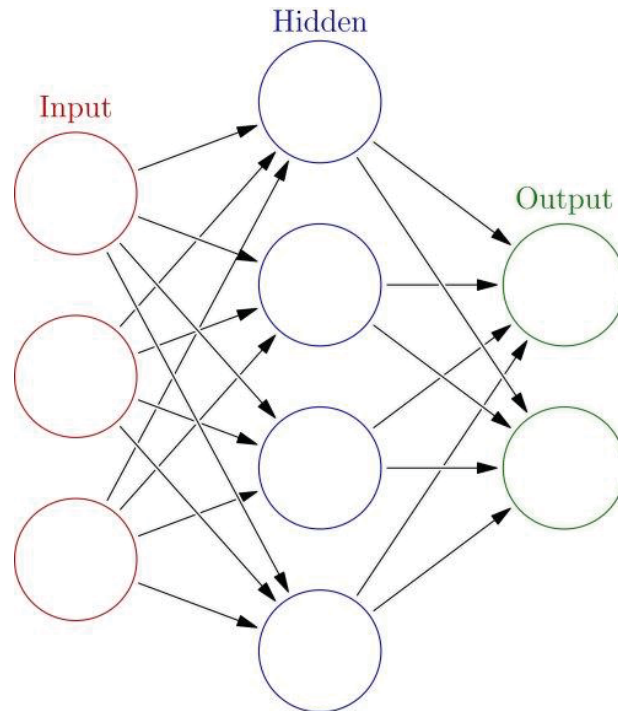


Figura 1-5: Capas del modelo Red Neuronal [13].

Se pueden distinguir 5 elementos que influyen en la activación de los nodos, dichos elementos se presentan en la Figura 1-6.

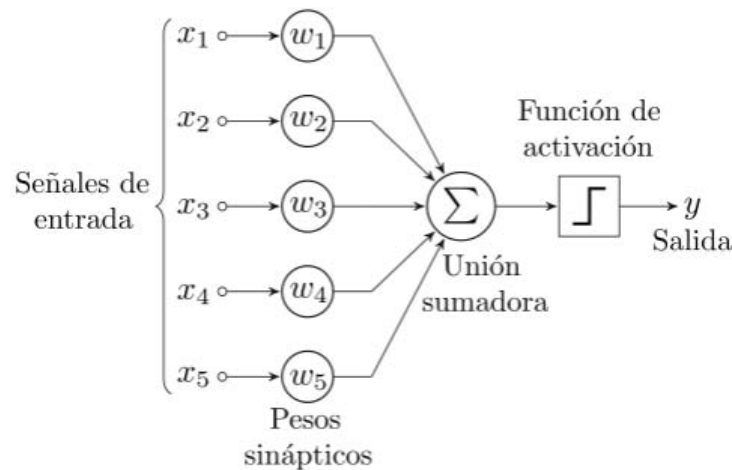


Figura 1-6: Esquema de un nodo del modelo Redes Neuronales [14].

Las señales de entrada es la información que recibe de todas las neuronas de la capa anterior y que interactúan con el nodo en cuestión. Los pesos sinápticos son factores que ponderan a las señales de entrada, es el mismo modelo que reajusta dichos valores con el entrenamiento (aunque en algunos casos puede permanecer fijo ya sea por el usuario o por alguna configuración previa). La unión sumadora, tal como su nombre lo dice, suma cada una de las entradas ponderadas por los pesos sinápticos y llega a un resultado. La función de activación (decidida por el usuario) es el umbral de comparación, si el resultado de la unión sumadora es superior entonces se activa el nodo [15].

1.6 Minería de Datos

La Minería de Datos (también Exploración de Datos), es una ciencia de la computación que relaciona la estadística, Machine Learning, Big Data e Inteligencia Artificial, su objetivo consta de extraer información implícita y/o desconocida que posea un grupo de datos [16]. El proceso de la minería de datos se puede resumir en los siguientes pasos:

- **Conjunto de datos:** Se debe contar con un grupo de datos los cuales se tenga el mismo tipo de información para poder hacer comparaciones.
- **Selección de características:** Se debe dejar en claro cuáles son las variables objetivo (aquellas sobre las cuales se desea algún tipo de predicción) y variables independientes (aquellas sobre las que se efectúa algún cálculo o proceso).
- **Selección del modelo:** Se selecciona el tipo de modelo adecuado a la situación, ya sea predictivo, clasificación o regresión. Para este caso se elegirá un modelo de clasificación, más específicamente un bosque de decisiones.
- **Entrenamiento del clasificador:** Dependiendo del problema, aplica un tipo de aprendizaje (supervisado, no supervisado, por refuerzo) específico al clasificador.

- **Evaluación del clasificador:** Se comprueba el desempeño del sistema para saber si es posible fiarse de los resultados. La herramienta usada para la medición del desempeño de las aplicaciones creadas es la Matriz de Confusión.

A continuación, en la Figura 1-7 se muestra un esquema que muestra los pasos a seguir en la Minería de Datos.

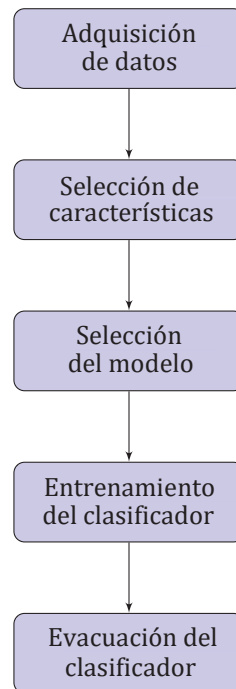


Figura 1-7: Esquema secuencial de la Minería de Datos.

1.6.1 Aplicaciones de la Minería de Datos

La Minería de Datos se puede aplicar en cualquier situación en la que se requiera saber algún tipo de tendencia, comportamiento o patrón que posea la información de algún sistema parcialmente conocido (si es totalmente conocido entonces no es necesaria su aplicación), también se requiere de una gran cantidad de datos para ser analizados (para así sacar provecho del método, de lo contrario se podrían aplicar otras alternativas más prácticas) y finalmente se requiere de software y hardware potentes para que sean capaces de procesar todo en una cantidad de tiempo prudente [17]. Algunos ejemplos de aplicación son:

- **Negocios:** Ayuda a la administración empresarial mostrando cómo se comportan sus clientes.
- **Fraudes:** Para detectar alguna anomalía con el uso de tarjetas de crédito o similares.
- **Genética:** Detectar cambios en la secuencia de ADN con el fin de detectar algún tipo de enfermedad.

- **Antiterrorismo:** Agencias de inteligencia analizan eventos ocurridos para identificar grupos terroristas y sus objetivos.

- **Recursos Humanos:** Sirve para la identificación de las características que identifican a los buenos empleados de una empresa.

1.7 Trabajo a realizar

Con el fin de comprobar los beneficios de las plataformas de Cloud Computing, se procede a crear una cuenta para utilizar los servicios de Microsoft Azure Studio, Amazon Web Services y Google Cloud Platform. Para esta ocasión, se pretende recrear las aplicaciones mencionadas en un artículo de una revista [18], los cuales son: el ejemplo ilustrativo (aplicación etiquetadora de datos) y clasificador de imágenes Thomson Scattering.

Para este trabajo, se dispuso de las mismas bases de datos del artículo mencionado anteriormente, aunque los algoritmos para el procesamiento de la información son diferentes, ya que éstos fueron seleccionados y ajustados según cada plataforma virtual (cada plataforma ofrece sus propias herramientas de trabajo).

1.7.1 Etiquetador de Datos (Ejemplo ilustrativo)

A modo de experimento sencillo, se planea crear una aplicación que identifique y aprenda comportamientos implícitos de una determinada base de datos, todo esto con el objetivo de etiquetar datos futuros que carezcan de variable objetivo (característica sobre la cual se hacen las predicciones). La base de datos con la que se cuenta es una matriz en formato “.CSV” donde cada columna es una característica y cada fila corresponde a un elemento. A continuación, en la Tabla 1-2 se muestra la base de datos utilizada para la aplicación etiquetadora de datos.

Tabla 1-2: Base de Datos del “Etiquetador de Datos”.

X1	x2	clase
1	4	circle
0	5	circle
1	1	circle
2	1	circle
5	5	cross
5	4	cross
1,5	0,5	cross

Se sabe que los encabezados mostrados en la Tabla 1-2 (“X1”, “x2” y “clase”), corresponden a las características de la base de datos. Por lo que al graficar esta información en un plano cartesiano (gráfico mostrado en la Figura 1-8), se tiene a la característica “X1” en el eje de las abscisas, la característica “x2” en el eje de las ordenadas y por último la característica “clase”, que es sobre la cual se realizarán las predicciones, es por ello que se le denominará como “la variable objetivo”,

y se representa en el gráfico de la Figura 1-8 como etiqueta de cada coordenada (circle o cross según sea el caso).

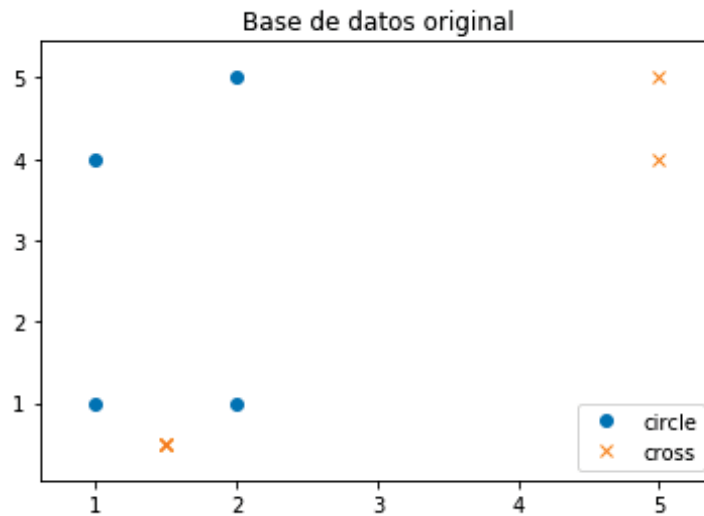


Figura 1-8: Base de Datos del “etiquetador de datos” graficada en el eje cartesiano.

Para implementar este sistema, se propone un esquema lógico a seguir, el cual enumera y ordena las tareas que deben cumplirse para realizar la aplicación, dicho esquema se muestra en la Figura 1-9

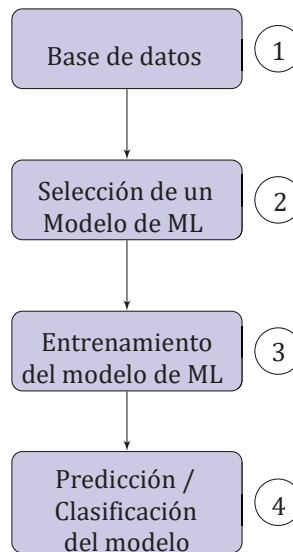


Figura 1-9: Diagrama de flujo del “etiquetador de datos”.

1.7.2 Clasificador de imágenes Thomson

Para esta segunda experiencia, se requiere de una aplicación capaz de reconocer y clasificar correctamente entre diferentes tipos imágenes de un proyecto de fusión nuclear, donde el tipo de imagen corresponde a la variable objetivo. La base de datos con la que se cuenta es una matriz

numérica en formato “CSV” donde cada columna representa a una imagen y cada fila representa el contenido de un pixel, por lo que para poder recrear una imagen es necesario utilizar códigos de programación. A continuación, en la Figura 1-10 se observan las distintas clases de imágenes que se deben diferenciar que se consiguen del Stellarator (dispositivo para confinar plasmas calientes mediante campos magnéticos con el objetivo de mantener reacciones de fusión nuclear controlada):

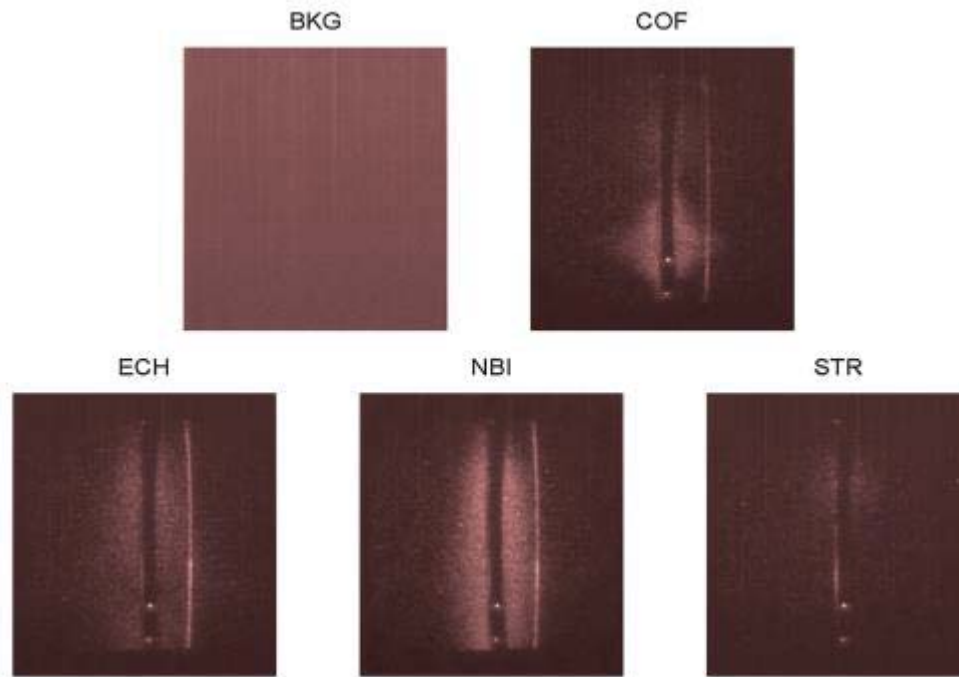


Figura 1-10: Recreación de las clases de imágenes del proyecto Thomson [19].

Donde los tipos de imágenes de la Figura 1-10 se explican a continuación en la Tabla 1-3 [19] [20].

Tabla 1-3: Tipos de imágenes Thomson Scattering

Tipo de imagen	Significado (inglés)	Significado (traducido)
BKG	CCD (Charge-Coupled Device) camera background.	Cámara del fondo sin luz.
COF	Cut-off density during electron cyclotron resonant heating.	Densidad de corte durante el calentamiento por resonancia de ciclotrón.
ECH	Electron cyclotron resonant heating.	Calentamiento por resonancia ciclotrónica electrónica.
NBI	Neutral beam injection heating.	Inyección de haces neutros.
STR	Stray light.	Luz parasita del sistema.

Cabe mencionar que las imágenes recreadas son de 576x385 píxeles cada una, en donde se utilizan dichos píxeles como discriminante para el reconocimiento de los distintos tipos de imagen, aunque no es buena alternativa ocupar cada píxel como una característica diferente ya que, tal cantidad de información demora considerablemente al sistema. Una alternativa para solucionar el problema del tiempo de demora, es reducir la cantidad de información de la base de datos original que se maneja, para ello hay técnicas de extracción de características con las cuales se puede obtener un nuevo conjunto de datos más pequeño que el original y manteniéndose fiel a éstos. En esta oportunidad, se generaron regiones (datasize) de 15x16 píxeles, para luego obtener un promedio, dicho valor representaba a todos los elementos que contenía el “datasize” por lo que debía mantener su posición en la imagen, los píxeles que no alcanzaban a agruparse de esta manera simplemente se omitían, por lo que las nuevas imágenes (y base de datos de entrada al modelo de entrenamiento) se muestran en la Figura 1-11.

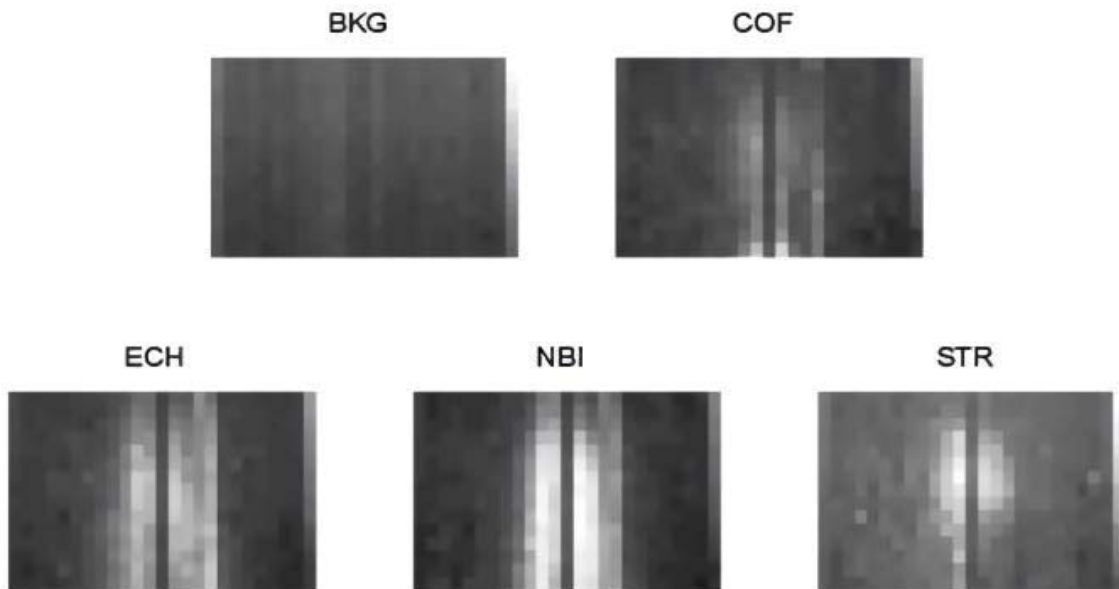


Figura 1-11: Recreación de las clases de imágenes del proyecto Thomson después de la reducción [20].

Al igual que en la experiencia anterior, se plantea el siguiente esquema de trabajo mostrado en la Figura 1-12, esto se hace con el mismo objetivo que el esquema de la Figura 1-9, ordenar las ideas y facilitar el entendimiento para la creación de la aplicación.

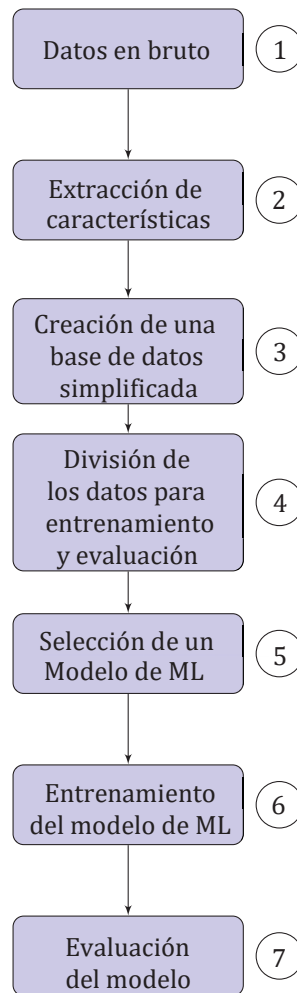


Figura 1-12: Diagrama de flujo para el clasificador de imágenes.

La cantidad de imágenes que se maneja se dividen en 2 grupos, el primer grupo que consta del 70% y se utiliza para el entrenamiento del modelo, luego, el 30% restante se utiliza para medir el desempeño del clasificador. El modelo será evaluado mediante una Matriz de Confusión, ya que permite observar de manera cómoda, fácil y compacta, los porcentajes de acierto y fallo del modelo en cuestión, sobre cada una de las clases de la variable objetivo. Para poder interpretar la información entregada por esta herramienta, se debe saber de antemano que las categorías como filas representan a los valores reales, mientras que las columnas corresponden a los valores predichos por el modelo, tal como se explica en la siguiente tabla.

Tabla 1-4: Matriz de Confusión.

		Valores predichos		
		clase 1	clase 2	clase 3
Valores reales	clase 1	A%	B%	C%
	clase 2	D%	E%	F%
	clase 3	X%	Y%	Z%

Donde “A%” de la Tabla 1-4, corresponde al porcentaje de la “clase 1” clasificada como “clase 1”, mientras que “B%” corresponde al porcentaje de la “clase1” clasificada como “clase 2” según el modelo de Machine Learning utilizado, cabe destacar que la suma de todos los valores de una misma fila debe ser de 100%, ya que eso indica lo ocurrido con todos los datos de una misma categoría.

Una vez descritas las experiencias a realizar, se procede a estudiar las plataformas virtuales con la intención de implementar las respectivas soluciones en ellas.

2 Microsoft Azure Studio

2.1 Microsoft Azure Studio (Cloud Computing)

La empresa Microsoft ofrece a disposición de sus clientes una herramienta llamada Azure Machine Learning Studio, la cual es una plataforma de trabajo interactivo (PaaS) usada para desarrollar aplicaciones que satisfagan necesidades específicas relacionadas con el reconocimiento de patrones, sin la necesidad de usar algún lenguaje de programación (aunque da la posibilidad de crear scripts hechos en Python o R), sino que simplemente conectados bloques que vienen pre-definidos en la plataforma. En la Figura 2-1, se observa un menú desplegable con las opciones (tanto de configuración como de trabajo) disponibles en Microsoft Azure Studio.

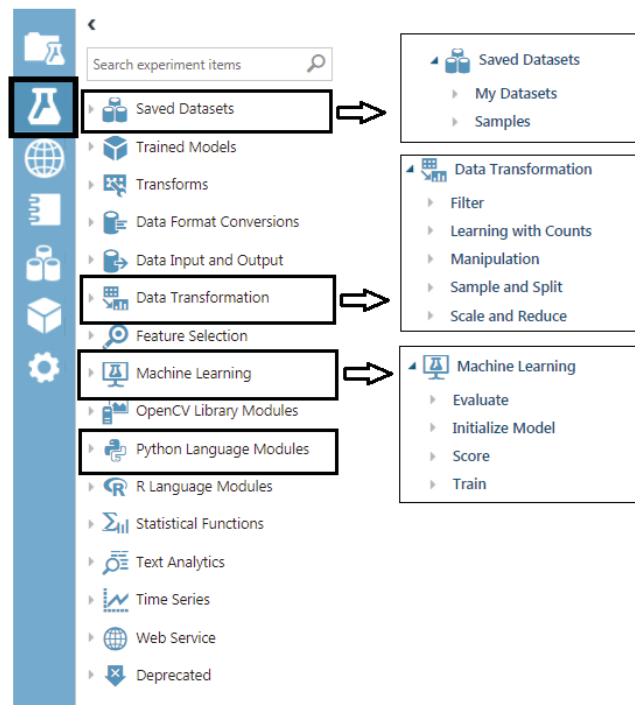


Figura 2-1: Herramientas de Microsoft Azure Studio.

Como se dijo antes de la Figura 2-1, en la primera columna se aprecian unos iconos, los cuales corresponden a lo siguiente (respetando el orden):

- **Projects:** Aquí se pueden almacenar experimentos, conjuntos de datos y demás archivos que se quieran ordenar de manera conjunta (análogo a guardar archivos dentro de una carpeta).
- **Experiments:** Aquí se almacenan y editan todos los experimentos creados por el usuario, siempre y cuando hayan sido previamente guardados.
- **Web Services:** Almacena los servicios webs que viene de los experimentos.
- **Notebooks:** Se tiene acceso a los cuadernos virtuales creados en Jupyter (ambiente de trabajo interactivo en línea, se usa para almacenar presentaciones, trabajos y códigos de programación).
- **Datasets:** Almacena conjuntos de datos que se han subido correctamente a la nube.
- **Trained Model:** Almacena los modelos que se han creado, entrenado y guardado en ocasiones anteriores.
- **Setting:** Tiene las configuraciones de cuenta y recursos.

Al ingresar en la opción “Experiment” de la plataforma (la única opción destacada en la primera columna de la Figura 2-1), se despliegan todas las alternativas de Machine Learning ofrecidas por Microsoft Azure Studio, las cuales están mostradas en la segunda columna de la Figura 2-1, y contienen los bloques mostrados en la última columna de la misma figura, dichos bloques son los usados para el desarrollo de las aplicaciones realizadas en esta plataforma virtual.

Para la creación de cualquier experimento en Azure Studio, basta con buscar el bloque necesario, para luego arrastrarlo y dejarlo la zona de trabajo, donde finalmente se conectan de diferente forma según su función (la misma plataforma indica las entradas y salidas de cada bloque), para que interactúen entre sí y cumplan su objetivo [21].

2.2 Etiquetador de Datos en Microsoft Azure Studio

Habiendo mencionado previamente como usar Microsoft Azure Studio para la creación de una aplicación, se procede a trabajar en el programa “Etiquetador de Datos” mencionados en el Capítulo 1, lo que da como resultado la programación en bloques mostrada en la Figura 2-2.

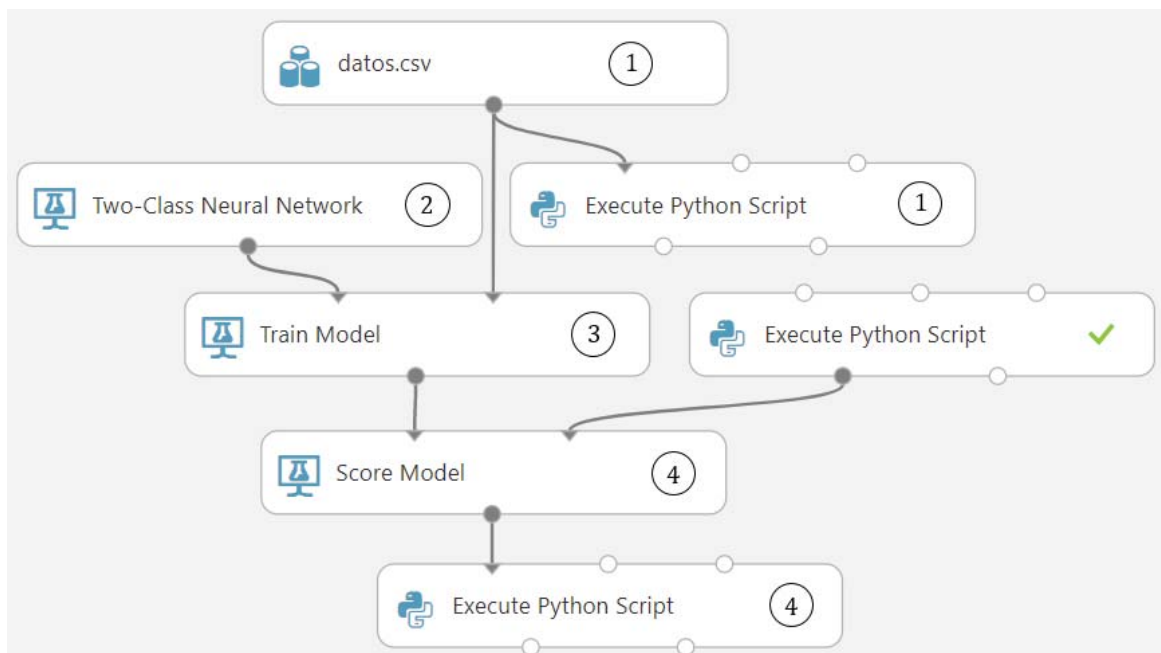


Figura 2-2: Programación gráfica del etiquetador de clases.

Se observa que mantiene el esquema planteado en la Figura 1-9, esto se logra apreciar de mejor manera al reordenar y agrupar de distinta manera los bloques, tal como se muestra en a Figura 2-2. Luego, con un script de Python, se grafican los resultados obtenidos por el clasificador, pudiendo etiquetar datos nuevos que carecían de clase, dando como resultado la Figura 2-3.

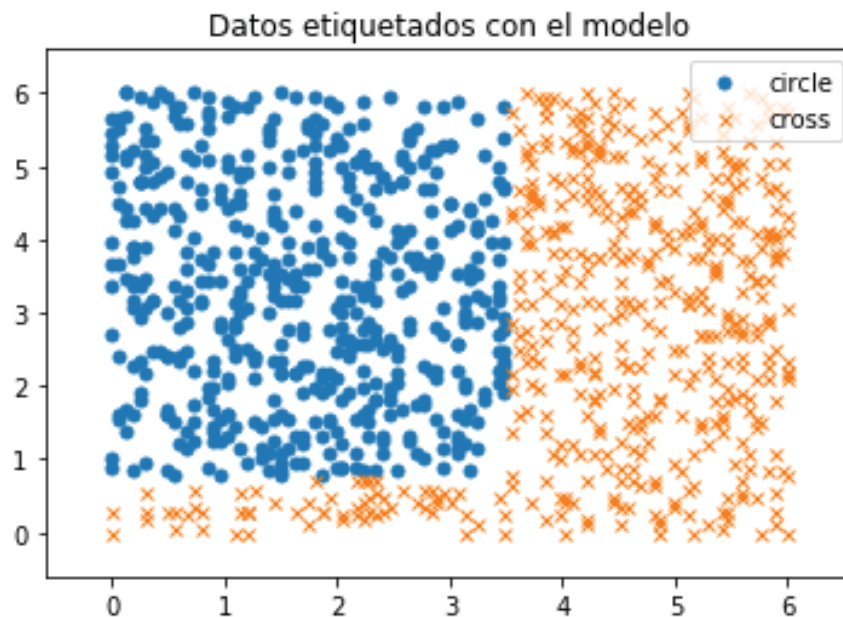


Figura 2-3: Resultado del etiquetador de datos hecho en Azure Studio.

2.3 Clasificador de Imágenes Thomson en Microsoft Azure Studio

Ahora es turno de desarrollar la aplicación “Clasificador de Imágenes Thomson”, donde análogo al caso anterior, se diseña una aplicación de Machine Learning utilizando el entorno gráfico de Azure Studio, y siguiendo el esquema correspondiente a la Figura 1-12, se obtiene como resultado la siguiente programación en bloque mostrada en la Figura 2-4.

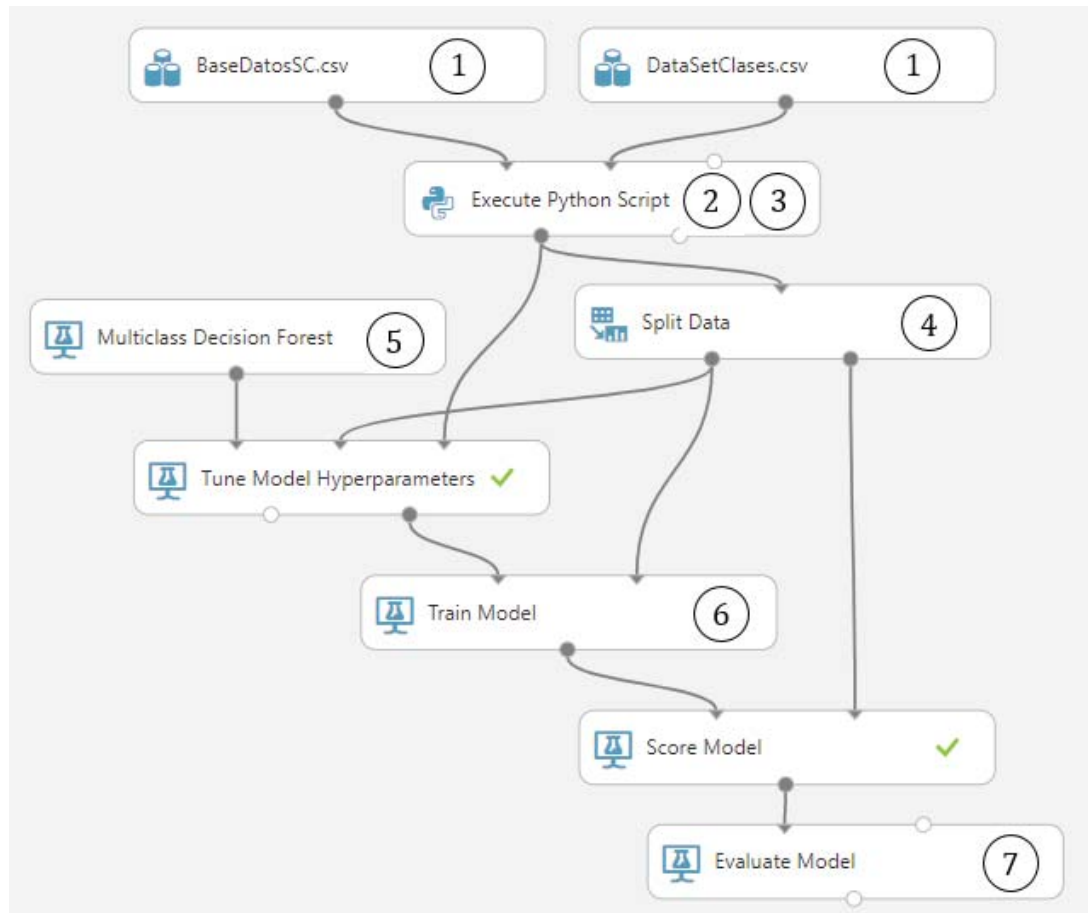


Figura 2-4: Programación gráfica del clasificador de imágenes

Para completar la aplicación, se debe agregar un elemento evaluador, esto con el fin de cuantificar desempeño del sistema. Luego, se comprueban distintos modelos ofrecidos por la plataforma virtual, para ello se utiliza el mismo esquema mostrado en la figura 2-4 (con la diferencia que se va cambiando el bloque de Machine Learning según el modelo que se desee utilizar). A continuación, se muestran las Matrices de Confusión de los distintos modelos utilizados en los experimentos realizados en Microsoft Azure Studio.

- **Modelo Bosque de Decisiones:**

Tabla 2-1: Matriz de confusión del modelo Boque de Decisiones hecho en Azure Studio.

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	89%	0%	0%	11%
ECR	0%	0%	93%	0%	0%
NBI	0%	0%	10%	90%	0%
STR	0%	5%	0%	0%	95%

En la Tabla 2-1, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Bosque de Decisiones). Se observa una correcta clasificación para la clase BKG, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clasificación de la clase COF, de la cual se tiene un 89% de acierto.

- **Modelo Uno vs Todos con Árbol de Decisiones:**

Tabla 2-2: Matriz de Confusión del Modelo Uno vs Todos con árbol de decisiones en Azure Studio.

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	86%	0%	5%	9%
ECR	0%	0%	100%	0%	0%
NBI	0%	0%	6%	94%	0%
STR	4%	0%	4%	0%	91%

En la Tabla 2-2, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Uno vs Todos con Árbol de Decisiones). Se observa una correcta clasificación para las clases BKG y ECH, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clasificación de la clase COF, de la cual se tiene un 86% de acierto.

- **Modelo Jungla de Decisiones:**

Tabla 2-3: Matriz de Confusión del modelo jungla de decisiones hecho en Azure Studio.

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	94%	0%	0%	6%
ECR	0%	0%	100%	0%	0%
NBI	0%	0%	6%	100%	0%
STR	0%	5%	5%	0%	90%

En la Tabla 2-3, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Jungla de Decisiones). Se observa una correcta clasificación para las clases

BKG, ECH y NBI, aunque se notan problemas para distinguir las otras dos categorías restantes, sobre todo para la clasificación de la clase STR, de la cual se tiene un 90% de acierto.

- **Modelo Regresión Logística Multinomial:**

Tabla 2-4: Matriz de confusión Regresión Logística Multinomial hecho en Azure Studio

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	89%	0%	0%	11%
ECR	0%	0%	100%	0%	0%
NBI	0%	0%	0%	100%	0%
STR	0%	0%	0%	0%	100%

En la Tabla 2-4, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Regresión Logística Multinomial). Se observa una correcta clasificación para las clases BKG, ECH, NBI y STR, aunque se notan problemas para distinguir de la manera deseada a la clase COF, de la cual se tiene un 89% de acierto.

A modo de experimento, se decide cambiar el tamaño del dataseize establecido anteriormente a uno más pequeño que sería de 8x7, el cual fue escogido de manera arbitraria con el objetivo de analizar y comparar resultados para posteriormente obtener conclusiones, por lo que se obtienen los siguientes resultados:

- **Modelo Bosque de Decisiones:**

Tabla 2-5: Matriz de confusión del modelo Boque de Decisiones hecho en Azure Studio con dataseize de 8x7

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	89%	0%	0%	11%
ECR	0%	0%	100%	0%	5%
NBI	0%	0%	10%	90%	0%
STR	0%	5%	0%	0%	95%

En la Tabla 2-5, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Bosque de Decisiones). Se observa una correcta clasificación para las clases BKG, ECH, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clase COF, de la cual se tiene un 89% de acierto.

- **Modelo Uno vs Todos con Árbol de Decisiones:**

Tabla 2-6: Matriz de Confusión del modelo Uno vs Todos con árbol de decisiones en Azure Studio con datasize de 8x7

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	82%	0%	0%	18%
ECR	0%	0%	100%	0%	0%
NBI	0%	0%	6%	94%	0%
STR	4%	4%	13%	0%	79%

En la Tabla 2-6, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Uno vs Todos con Árbol de Decisiones). Se observa una correcta clasificación para las clases BKG, ECH, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clase STR, de la cual se tiene un 79% de acierto.

- **Modelo Jungla de Decisiones:**

Tabla 2-7: Matriz de confusión del modelo Jungla de Decisiones hecho en Azure Studio con datasize de 8x7

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	100%	0%	0%	0%
ECR	0%	0%	100%	0%	0%
NBI	0%	0%	10%	90%	0%
STR	0%	0%	0%	0%	100%

En la Tabla 2-5, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Bosque de Decisiones). Se observa una correcta clasificación para las clases BKG, ECH, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clase COF, de la cual se tiene un 89% de acierto.

- **Modelo Regresión Logística Multinomial:**

Tabla 2-8: Matriz de confusión Regresión Logística Multinomial hecho en Azure Studio con datasize de 8x7

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
CUT	0%	89%	0%	0%	11%
ECR	0%	0%	100%	0%	0%
NBI	0%	0%	10%	90%	0%
STR	0%	0%	0%	0%	100%

En la Tabla 2-8, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Regresión Logística Multinomial). Se observa una correcta clasificación para las clases BKG, ECH y STR, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clase COF, de la cual se tiene un 89% de acierto.

3 Amazon Web Services (AWS)

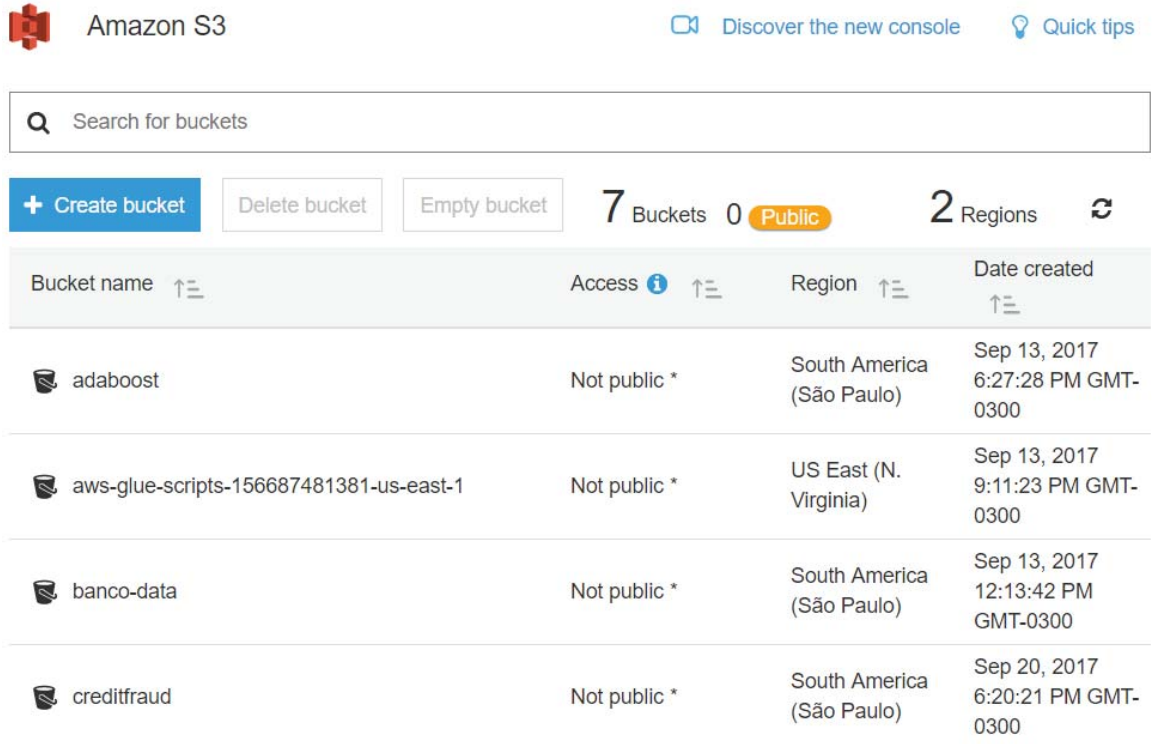
3.1 Amazon Web Services (Cloud Computing)

Es una empresa que ofrece un amplio conjunto en servicios de Cloud Computing, tales como: informática, almacenamiento de información, bases de datos, internet de las cosas, inteligencia artificial y desarrollo de juegos (entre otros). Un punto importante que cabe mencionar sobre Amazon Web Services, es el hecho de ser una de las empresas con mayor tiempo en la industria, junto con proporcionar servicios de fácil acceso (entiéndase por fácil acceso, a que ofrece en sus productos, interfaces en los cuales es relativamente sencillo de aprender y utilizar por los usuarios), dándoles como resultado una buena reputación a nivel mundial.

Para el desarrollo de los sistemas creados en la plataforma, solo se utilizaron dos de las tantas herramientas ofrecidas por AWS, la cuales son: “Amazon S3” para el almacenaje de información y “Amazon Machine Learning” para el aprendizaje automático.

3.1.1 Almacenamiento S3

Es un servicio de almacenamiento de objetos y todo tipo de datos, tiene la ventaja de ser la opción de almacenamiento con mayor compatibilidad con el resto de los productos de Amazon Web Services, es por ello que se elige este servicio.



The screenshot shows the Amazon S3 console interface. At the top, there's a search bar for buckets. Below it, there are buttons for 'Create bucket', 'Delete bucket', and 'Empty bucket'. A summary bar indicates '7 Buckets', '0 Public', and '2 Regions'. The main content is a table listing buckets with columns for Bucket name, Access, Region, and Date created.

Bucket name	Access	Region	Date created
adaboost	Not public *	South America (São Paulo)	Sep 13, 2017 6:27:28 PM GMT-0300
aws-glue-scripts-156687481381-us-east-1	Not public *	US East (N. Virginia)	Sep 13, 2017 9:11:23 PM GMT-0300
banco-data	Not public *	South America (São Paulo)	Sep 13, 2017 12:13:42 PM GMT-0300
creditfraud	Not public *	South America (São Paulo)	Sep 20, 2017 6:20:21 PM GMT-0300

Figura 3-1: Interfaz gráfica de S3 (almacenamiento de archivos en carpetas virtuales)

3.1.2 Amazon Machine Learning

Es un servicio que pone a disposición del desarrollador, las herramientas necesarias del aprendizaje automático, de forma tal, que hay necesidad de programar complicados algoritmos, es por ello que se elige este servicio. Existen 4 tipos de tareas con las que trabaja Amazon Machine Learning, las cuales son:

- **Datasource:** Hace referencia a la base de datos con la cual se entrenará el modelo de Machine Learning.
- **ML Model:** Indica que el archivo es un modelo de Machine Learning, Amazon selecciona de manera automática el modelo a utilizar dependiendo de la naturaleza de la variable objetivo.
- **Evaluation:** Es un archivo que muestra el desempeño del modelo, se genera una Matriz de Confusión para facilitar la lectura de los resultados.
- **Batch Prediction:** Son los resultados que se obtienen al realizar predicciones sobre nuevos datos con el sistema ya finalizado.

Estas tareas se llevan a cabo de manera secuencial (en el orden que fueron expuestas), donde se sigue las instrucciones que el mismo servicio da a modo de guía, los pasos a seguir para la creación de las aplicaciones (que a su vez, mantienen la esencia de las Figura 1-9 y Figura 1-12), tal como se muestran en la Figura 3-2.



Figura 3-2: Esquema de trabajo en Amazon Machine Learning.

- **Step 1:** Se carga una base de datos almacenada en S3, se pueden filtrar los datos para la creación de un archivo Datasource (datos de entrada al modelo de Machine Learning). En la Figura 3-3, se observa la interfaz gráfica en donde se carga una base de datos previamente almacenada en S3.

La interfaz muestra la siguiente estructura:

- Menú de navegación: Amazon Machine Learning > Datasources > Create datasource
- Progreso de los pasos: 1. **Input Data** | 2. Schema | 3. Target | 4. Row ID | 5. Review
- Título: Input data
- Descripción: Import your data to create an Amazon ML datasource. Amazon ML can use your datasource to create and evaluate an ML model, and you can use the datasource to review your data.
- Opciones de origen de datos: 'Where is your data?' con botones para S3 y Amazon Redshift.
- Sección 'S3 data access':
 - Texto: Tell Amazon ML how to access your data and give it permission to access it.
 - Campo 'S3 location *': s3:// bucket-name/file.csv
 - Nota: Enter the path to a single file or folder in Amazon S3. You need to grant Amazon ML permission to read this data. [Learn more.](#)
 - Nota: If you already have a schema for this data, provide it in a file at s3://<path-of-input-data>.schema. If you don't have a schema, Amazon ML will help you create one on the next page.
 - Campo 'Datasource name': []
 - Botones: * Required, Reset, Cancel, Verify.

Figura 3-3: Interfaz gráfica que ayuda a cargar información guardada en S3.

- **Step 2:** Se crea y entrena un modelo de Machine Learning (creación de ML Model), también se da la opción de evaluarlo una vez terminado (Evaluation). En la Figura 3-4 se observa la interfaz en la cual se seleccionan los datos para el entrenamiento del modelo.

Amazon Machine Learning ▾ ML models > Create ML model

1. Input data **2. ML model settings** 3. Recipe 4. Advanced settings 5. Evaluation 6. Review

ML model settings

You can use the automatically suggested ML model settings, or you can choose to customize.

ML model type MULTICLASS ⓘ

ML model target class

ML model name (Optional)

Select training and evaluation settings Recipes and training parameters control the ML model training process. You can select these settings for your ML model or use the defaults provided by Amazon ML. In either case, you can choose to have Amazon ML reserve a portion of the input data for evaluation. [Learn more.](#)

Default (Recommended)

- Generate a default recipe
- Use default training parameters
- Set aside 30% of your training data to evaluate the training
- Split the evaluation data sequentially ⓘ

Custom


- Modify the recipe Amazon ML generates
- Modify training parameters
- Randomly or sequentially split your evaluation data ⓘ

Evaluation Name

[Cancel](#)
[Previous](#)
[Review](#)

Figura 3-4: Interfaz gráfica para el entrenamiento del modelo en Amazon Machine Learning

- **Step 3:** Se utiliza el proyecto creado para realizar predicciones sobre nuevos datos (Batch Prediction). En la Figura 3-5, se observa la interfaz en la cual se selecciona un modelo ya entrenado para realizar predicciones.

 Amazon Machine Learning ▾

[Batch Predictions](#) > [Create batch prediction](#)

1. ML model for batch prediction 2. Data for batch prediction **3. Batch prediction results** 4. Review

Batch prediction results ?

The estimated cost for generating your predictions is **\$0.20**. This estimate is based on the size of the data in your prediction request. The accuracy of the estimate depends on the distribution and variance of your data records. Your cost may vary from this estimate.

Your data size is 10.3 KB. Amazon ML estimates the average data record size as 9 bytes. The estimated number of predictions is 1,176. [i](#)

The Amazon ML fee for batch predictions is **\$0.10 per 1,000 predictions**, rounded up to the next 1,000. [Learn more.](#)

Type the path to the S3 location in which the prediction results will be saved.

S3 destination

Batch prediction name (Optional)

Cancel
Previous
Review

Figura 3-5: Interfaz gráfica de Amazon Machine Learning para la predicción de la información

3.2 Etiquetador de Datos en Amazon Web Services

Se procede a trabajar en Amazon Machine Learning para el desarrollo de la aplicación etiquetadora de datos, pero primero se almacena en S3 la base de datos correspondiente a la Tabla 1-2, la cual se encuentra en formato .CSV, para a creación de un Datasource, luego se trabaja siguiendo las instrucciones de la herramienta respetando el orden mostrado en la Figura 3-6.

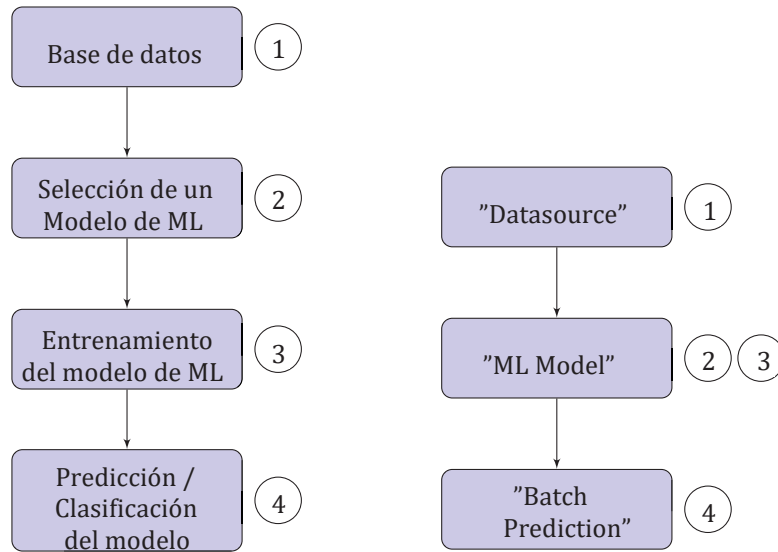


Figura 3-6: Comparación entre lo planificado y lo realizado en AWS con respecto al etiquetador

Luego de completar el procedimiento anterior, se obtiene como resultado un archivo .CSV en el cual está la predicción de la etiqueta correspondiente, por lo que utilizando programación en Python para poder observar de mejor manera los resultados, se obtiene el siguiente plano cartesiano mostrado en la Figura 3-7.

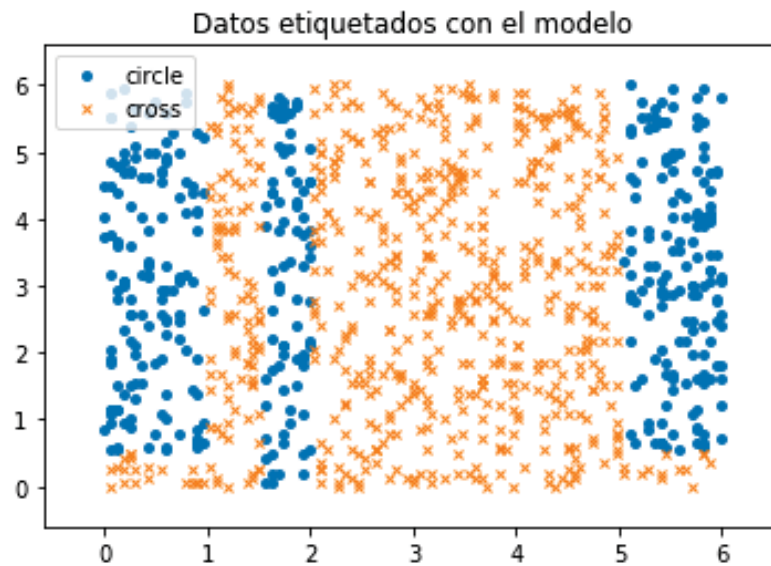


Figura 3-7: Grafico el resultado del etiquetador en AWS

3.3 Clasificador de Imágenes Thomson en Amazon Web Services

Una de las cualidades de Amazon Machine Learning, es la de mantener el mismo proceso de desarrollo de aplicaciones (indistintamente si se trata de un clasificador de 2 clases o multiclase

como lo son ambos experimentos), aunque para este caso la base de datos originales llamada “Datos en Bruto” podía ser utilizada directamente en la plataforma, entonces se utilizó un código de Python para combinar los dos archivos .CSV y realizar la extracción de características mencionadas en el Capítulo 1. Por lo que adaptándose al esquema de la Figura 1-9 en esta plataforma, se obtiene el siguiente nuevo esquema mostrado en la Figura 3-8.

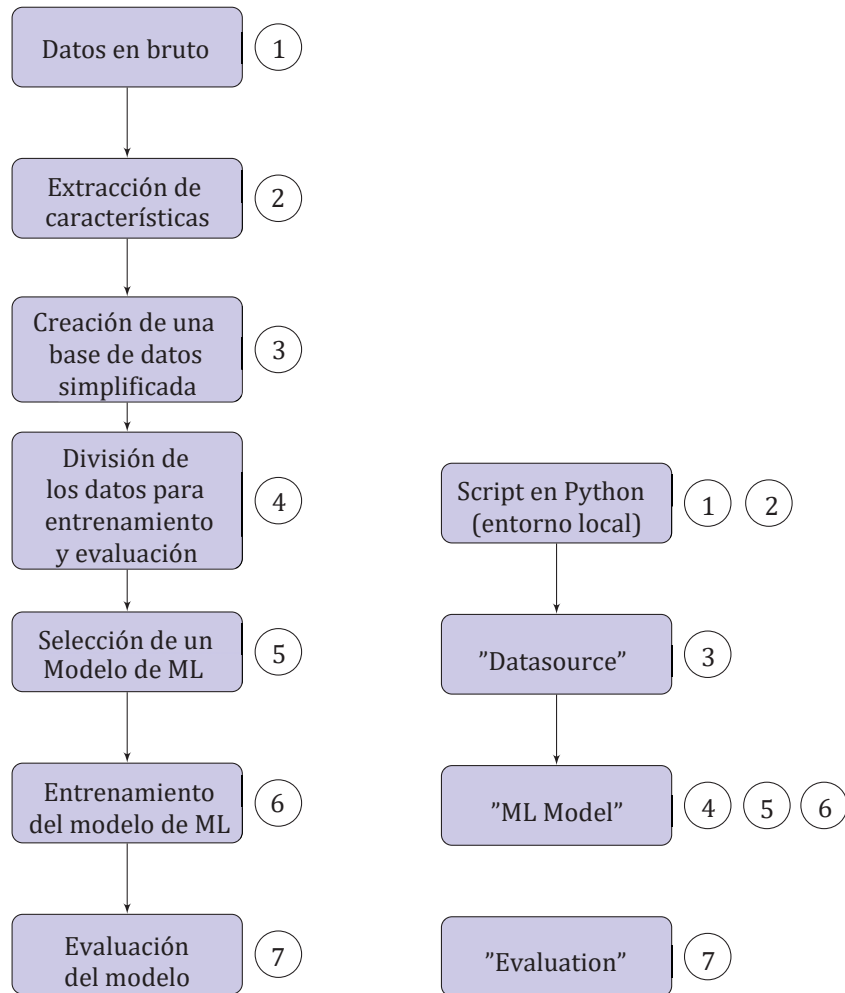


Figura 3-8: Comparación entre algoritmo y trabajo realizado del clasificador de imágenes en Amazon

Luego, se observa la Matriz de Confusión entregada por Amazon para medir el desempeño del modelo, teniendo en cuenta que la plataforma no permite cambio de modelo, es decir, el modelo se selecciona de manera automática dependiendo del tipo al que pertenece la variable objetivo.

- Modelo Regresión Logística Multinomial:

Tabla 3-1: Matriz de confusión Regresión Logística Multinomial hecho en Amazon Web Services

	BKG	COF	ECH	NBI	STR
BKG	83%	0%	0%	0%	17%
COF	0%	86%	0%	7%	7%
ECH	0%	8%	75%	17%	0%
NBI	0%	0%	0%	100%	0%
STR	14%	7%	14%	0%	64%

En la Tabla 3-1, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Regresión Logística Multinomial). Se observa una correcta clasificación para la clase NBI, aunque se notan problemas para distinguir el resto de las categorías, sobre todo para la clase STR, de la cual se tiene un 64% de acierto.

A modo de experimento, se intentó replicar la experiencia del clasificador de imágenes cambiando el conjunto de datos a agrupaciones de 8x7 píxeles, lamentablemente no fue posible concluir la aplicación debido a que la cantidad de características de los datos de entrada era mayor a la soportada por la herramienta virtual que ofrece Amazon.

4 Google Cloud Platform (GCP)

4.1 Google Cloud Platform (Cloud Computing)

Google Cloud Platform es la plataforma virtual de Google encargada de proveer servicios de Cloud Computing, dicha plataforma dispone de herramientas con funciones específicas que pueden interactuar entre sí para según las necesidades del usuario, similar a lo que ocurre con Amazon Web Services. Aunque los entornos virtuales de Amazon y Google funcionan de manera similar, este último es más complicado de utilizar, ya que se maneja una consola virtual a la cual se debe ir configurando y habilitando API según las herramientas a utilizar en el proyecto.

Algunos de los servicios de GCP son de pago (como el uso de Machine Learning) y el saldo se descuenta de manera automática de la tarjeta de crédito anexada a la cuenta, pero al momento de crearse una cuenta, Google automáticamente carga un saldo inicial de \$300 dólares con duración de 1 año para que el usuario pueda aprender a utilizar la plataforma realizando experimentos (con esto Google compensa la dificultad del uso de la plataforma).

Para el desarrollo de los clasificadores propuestos en el Capítulo 1, solo se utilizaron dos de las tantas herramientas ofrecidas por GCP, las cuales son: “Storage” para el almacenamiento de datos y “Compute Engine” para la creación de un computador virtual. Además, se necesita de una librería especial para trabajo de Machine Learning llamada Tensorflow, la cual fue desarrollada por Google, dicha librería permite crear el modelo y los ajustes deseados.

4.1.1 Almacenamiento en “Storage”

Similar a lo que ocurre con la herramienta de almacenamiento S3 de Amazon Web Services, Storage de Google Cloud Platform es el servicio de almacenamiento de información que es compatible con la mayor cantidad de aplicaciones, de las cuales se encuentra Compute Engine. La interfaz gráfica de Storage se muestra en la Figura 4-1, donde se puede ver el uso y creación de elementos llamados segmentos, los cuales son carpetas con nombres únicos que sirven para guardar y ordenar la información (tanto base de datos como códigos de programación) de un proyecto.

Navegador CREAR UN SEGMENTO ↻ 🗑️ MOSTRAR PANEL DE INFORMACIÓN

Columnas ▾

Segmentos

<input type="checkbox"/> Nombre	Clase de almacenamiento predeterminada ?	Ubicación	Ciclo de vida ?	Etiquetas ?
<input type="checkbox"/> adaboost	Regional	SOUTHAMERICA-EAST1	Ninguna	⋮
<input type="checkbox"/> api-predictiva	Regional	SOUTHAMERICA-EAST1	Ninguna	⋮
<input type="checkbox"/> api_censo	Regional	SOUTHAMERICA-EAST1	Ninguna	⋮
<input type="checkbox"/> clasificador-flores	Regional	SOUTHAMERICA-EAST1	Ninguna	⋮

Figura 4-1: Interfaz gráfica de Storage en Google Cloud Platform

4.1.2 Compute Engine

Esta herramienta de Google, ofrece al usuario la oportunidad de crear sus propios computadores virtuales llamados Instancias, dichas Instancias tienen especificaciones totalmente ajustables por el cliente, tales como: sistema operativo, disco duro, memoria RAM, entre otras opciones. Cabe mencionar que este servicio no es gratuito, se paga por la cantidad de minutos (redondeando hacia arriba) que se utiliza la Instancia, la tarifa varía dependiendo de las características de la máquina virtual creada. Todo lo mencionado anteriormente sobre Compute Engine (características de la Instancia y precio), puede verse en la Figura 4-2.

Nombre [?]

Zona [?]

Tipo de máquina [?]
Para seleccionar los núcleos, la memoria y las GPU, haz clic en Personalizar.


1 vCPU 3,75 GB de memoria [Personalizar](#)

[Actualiza la cuenta](#) para crear instancias con un máximo de 64 núcleos

Contenedor [?]

Desplegar una imagen de contenedor en esta instancia de VM. [Más información](#)

Disco de arranque [?]

 Nuevo disco persistente estándar de 10 GB
Imagen
Debian GNU/Linux 9 (stretch) [Cambiar](#)

Identidad y acceso de API [?]

Cuenta de servicio [?]

Alcance del acceso [?]

Permitir el acceso predeterminado
 Permitir el acceso completo a todas las API de Cloud
 Definir acceso para cada API

Cortafuegos [?]
Añade reglas de cortafuegos y etiquetas para permitir tráfico de red concreto de Internet

Permitir el tráfico HTTP
 Permitir el tráfico HTTPS

[Administración, discos, redes, claves SSH](#)

Si tienes créditos de la versión de prueba gratuita, se utilizarán para esta instancia

[Crear](#) [Cancelar](#)

24,67 \$ al mes (estimación)
Tarifa por horas efectiva: 0,034 \$ (730 horas al mes)

Elemento	Costes estimados
1 vCPU con 3,75 GB de memoria	34,67 \$/mes
Disco persistente estándar de 10 GB	0,40 \$/mes
Descuento por uso continuado [?]	- 10,40 \$/mes
Total	24,67 \$/mes

[Precios de Compute Engine](#) [?]

[Menos](#)

Figura 4-2: Interfaz para la creación de una instancia

Para la creación de cualquier aplicación, se deben tener en cuenta los siguientes puntos:

- Una base de datos.
- Un modelo de Machine Learning hecho localmente con Tensorflow.
- Ambos archivos (base de datos y modelo) almacenados en la nube (Storage).
- Crear una Instancia en Compute Engine.
- Utilizar la Instancia para procesar los datos con el modelo.
- Guardar resultados en la nube (Storage).

Para lograr hacer algunas tareas importantes en la Instancia, se deber ejecutar comando en la consola que muestra (es la única forma de realizar tareas). A continuación, en la Tabla 4-1 se

muestran las acciones ejecutadas en la Instancia, junto con su respectivo comando de Linux para ejecutarla.

Acción	Comando de Linux
Crear carpeta	\$mkdir "nombre de la carpeta a crear"
Copiar carpeta	\$gsutil cp -r gs://"dirección de la carpeta a copiar" / ./"destino de pegado"
Ejecutar un script de Python	\$python "nombre del archivo".py

Tabla 4-1: Acciones y comandos de Linux usados en la Instancia

4.2 Etiquetador de Datos en Google Cloud Platform

En esta ocasión se logró programar y validar (que es lo recomendado por Google Cloud Platform) de manera local el modelo Red Neuronal convolucional, simple y Fully Connected (todas las neuronas de una capa están conectadas a cada una de la capa siguiente), cuya función de activación es la tangente hiperbólica y cuyo código se encuentra en los anexos. Al seguir el esquema de trabajo planteado en las plataformas anteriores, se obtiene la Figura 4-3.

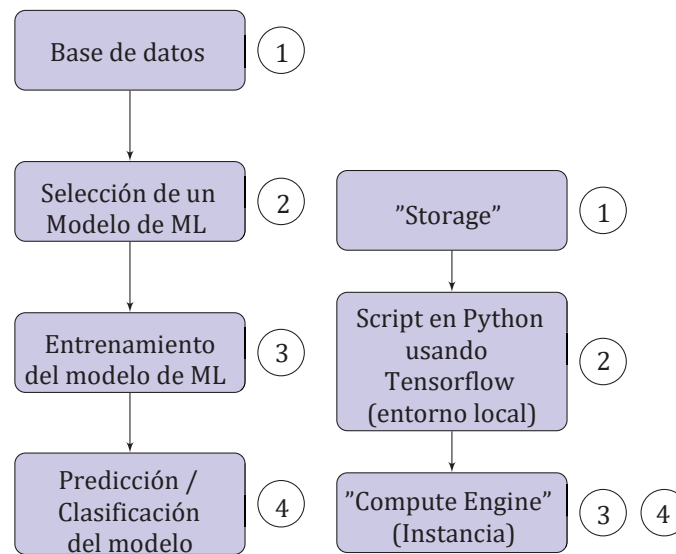


Figura 4-3: Esquema de trabajo para el etiquetador de clases en Google Cloud Platform.

Siguiendo el esquema de trabajo planteado en la Figura 4-3, se obtienen los resultados mostrados en la Figura 4-4.

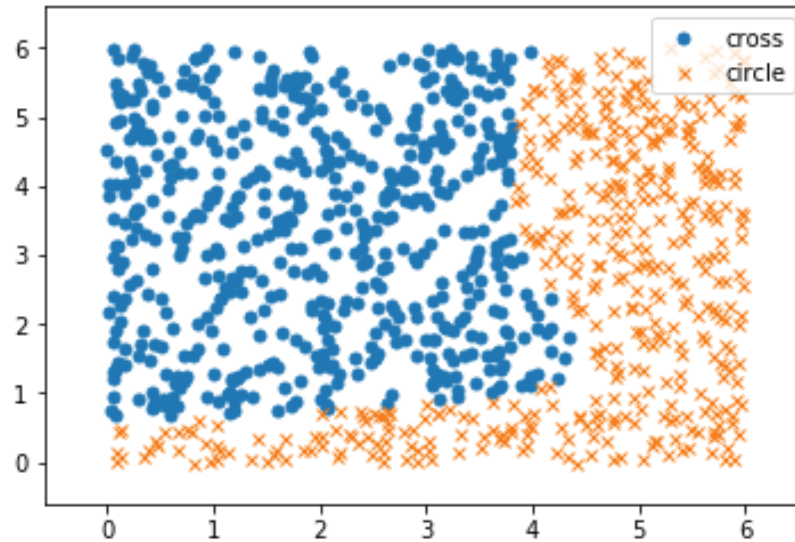


Figura 4-4: Resultado del etiquetador de clases obtenido en una Instancia de Google Cloud Platform.

4.3 Clasificador de Imágenes Thomson en Google Cloud Platform

Tal como se siguió un esquema de trabajo para la creación del etiquetador de clases, se pretende crear un esquema de trabajo que cumpla la pauta mostrada en la Figura 1-12, dando origen al esquema mostrado en la Figura 4-5.

Al seguir el esquema de trabajo propuesto, se obtiene la siguiente Matriz de Confusión mostrada en la Tabla 4-2

Tabla 4-2: Matriz de Confusión Redes Neuronales (datasize 15x16 pixeles)

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
COF	0%	92%	0%	8%	0%
ECH	0%	0%	100%	0%	0%
NBI	0%	7%	0%	93%	0%
STR	0%	0%	0%	0%	100%

En la Tabla 4-2, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Redes Neuronales). Se observa una correcta clasificación para las clases BKG, ECH y STR, aunque se notan ligeros problemas para distinguir el resto de las categorías, sobre todo para la clase COF, de la cual se tiene un 92% de acierto.

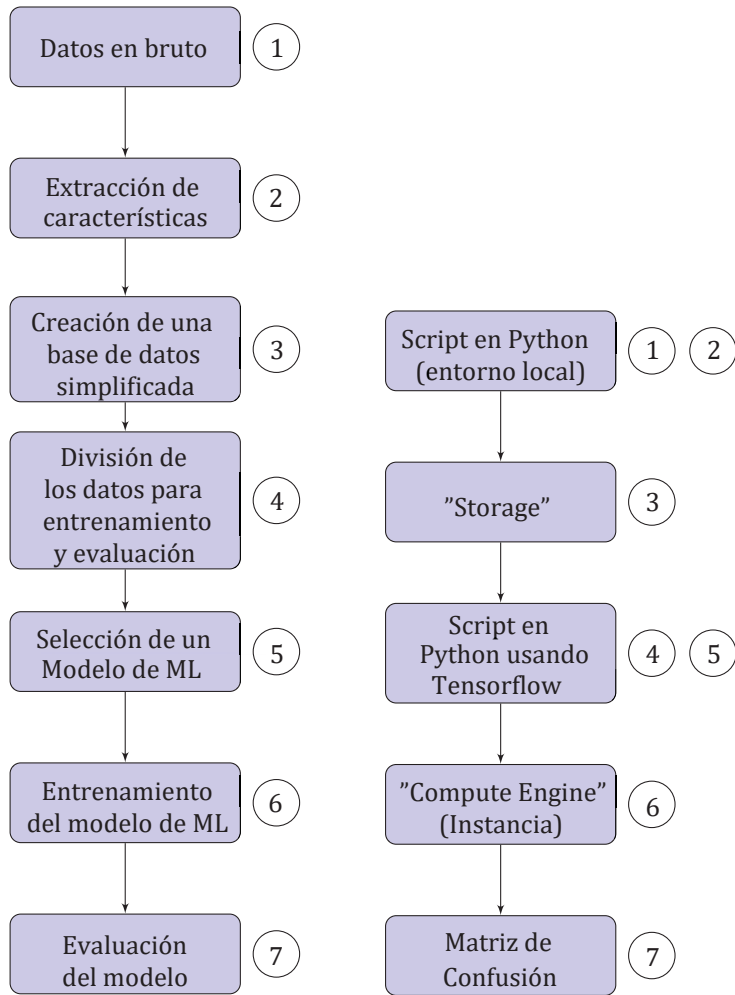


Figura 4-5: Esquema de trabajo para el clasificador de imágenes Thomson en Google Cloud Platform.

Al igual que se hizo en Microsoft Azure Studio, se decide cambiar el datasize planteado en el problema original por tamaños de 8x7 pixeles (manteniendo la misma base de datos y el mismo modelo), con el objetivo de ver como varían los resultados al cambiar levemente la técnica de extracción de características usada. Al hacerlo, se llega a la siguiente Matriz de Confusión mostrada en la Tabla 4-3.

Tabla 4-3: Matriz de Confusión Redes Neuronales (datasize 8x7 pixeles)

	BKG	COF	ECH	NBI	STR
BKG	100%	0%	0%	0%	0%
COF	0%	100%	0%	8%	0%
ECH	0%	0%	100%	0%	0%
NBI	0%	0%	0%	93%	7%
STR	0%	0%	0%	8%	92%

En la Tabla 4-3, se observa la Matriz de Confusión obtenida como resultado de la evaluación del modelo utilizado (Redes Neuronales). Se observa una correcta clasificación para las clases BKG, COF y ECH, aunque se notan algunos problemas para distinguir el resto de las categorías, sobre todo para la clase STR, de la cual se tiene un 92% de acierto.

5 Comparación de las plataformas

5.1 Comparación cuantitativa de los resultados obtenidos en el clasificador de imágenes Thomson.

5.1.1 Datasize de 15x16 pixeles.

Luego de haber estudiado y trabajado en las plataformas de interés, se procede a destacar y comparar los resultados obtenidos del clasificador de imágenes Thomson, dando origen a la Tabla 5-1.

Tabla 5-1: Resumen de los porcentajes de acierto de (datasize de 16x15)

	BKG	CUT	ECR	NBI	STR
Bosque de Decisiones (MAS)	100%	89%	93%	90%	95%
Uno vs Todos (Árbol) (MAS)	100%	86%	100%	94%	90%
Jungla de Decisiones (MAS)	100%	94%	100%	100%	90%
Regresión Logística Multinomial (MAS)	100%	89%	100%	100%	100%
Regresión Logística Multinomial (AWS)	100%	86%	64%	75%	83%
Redes Neuronales (GCP)	93%	92%	100%	100%	100%

A partir de los resultados mostrados en la Tabla 5-1, se propone promediar los porcentajes de acierto de cada uno de los experimentos realizados con el fin de tener algún criterio de comparación. A continuación, en la Tabla 5-2 se muestran los promedios mencionados.

Tabla 5-2: Promedio de porcentajes de aciertos de los experimentos (datasize de 15x16)

	Promedio de aciertos
Bosque de Decisiones (MAS)	93,4%
Uno vs Todos (Árbol) (MAS)	94%
Jungla de Decisiones (MAS)	96,8%
Regresión Logística Multinomial (MAS)	97,8%
Regresión Logística Multinomial (AWS)	81,6%
Redes Neuronales (GCP)	97%

En la Figura 5-2, se observa que el modelo Regresión Logística Multinomial implementada en la plataforma Microsoft Azure Studio (MAS) obtiene mejores resultados que el resto de los modelos, gracias a eso se pensaría que dicho modelo es el mejor para este problema en concreto, pero los resultados de la implementación hecha en Amazon Web Services hace pensar lo contrario, eso demuestra la importancia de los factores aleatorios y el correcto ajuste de los parámetros del modelo pueden hacer la diferencia entre un buen y mal modelo

5.1.2 Datasize de 8x7 pixeles.

A modo de experimento, se repite la experiencia de clasificador de imágenes Thomson, pero esta vez con un datasize menor (agrupaciones de 8x7 pixeles), lo que aumenta la cantidad de características que debe procesar el modelo a la hora de entrenarse el sistema, dando como resultado la Tabla 5-3 (como nota, se menciona el hecho de no poder replicar esta experiencia en la plataforma de Amazon, debido a que la cantidad de características generadas en esta ocasión superan a las permitidas en la plataforma).

Tabla 5-3: Resultados del clasificador de imágenes con un datasize de 8x7

	BKG	CUT	ECR	NBI	STR
Bosque de Decisiones (MAS)	100%	89%	100%	90%	95%
Uno vs Todos con Árbol de Decisiones (MAS)	100%	82%	100%	93%	78%
Jungla de decisiones (MAS)	100%	100%	100%	90%	100%
Regresión Logística Multinomial (MAS)	100%	89%	100%	90%	100%
Redes Neuronales (GCP)	93%	100%	92%	100%	100%

Ídem al caso anterior, a partir de los resultados mostrados en la Tabla 5-3, se obtienen los promedios de los porcentajes de acierto para cada modelo, generando así a la Tabla 5-4.

Tabla 5-4: Promedio de porcentajes de aciertos de los experimentos (datasize de 8x7)

	Promedio de aciertos
Bosque de Decisiones (MAS)	94,8%
Uno vs Todos con Árbol de Decisiones (MAS)	90,6%
Jungla de decisiones (MAS)	98%
Regresión Logística Multinomial(MAS)	95,8%
Redes Neuronales (GCP)	97%

Al comparar los resultados de las Tablas 5-2 y 5-4, se observa que el mejor caso se obtiene con el modelo de Jungla de Decisiones con un datasize de 8x7 pixeles y que el peor caso (sin contar con el modelo de Regresión Logística Multinomial de Amazon Web Services) se obtiene usando el mismo datasize con el modelo Uno vs Todos con Árbol de Decisiones.

5.2 Comparación cualitativa entre las plataformas

Gracias al trabajo realizado en cada una de las plataformas virtuales, fue posible identificar y comprobar de manera empírica, algunas de características que resaltan de estas plataformas virtuales. A continuación, en la Tabla 5-5 se presenta una comparación de los diferentes servicios de Cloud Computing utilizados durante el proyecto.

Tabla 5-5: Comparación de las distintas plataformas de Cloud Computing estudiadas en el proyecto

Microsoft Azure Studio	Amazon Web Services	Google Cloud Platform
Servicios básicos de Machine Learning se encuentran en la capa gratuita.	Aunque algunos de sus servicios se encuentran en la capa gratuita, se debe pagar por el uso de Machine Learning.	Aunque algunos de sus servicios se encuentran en la capa gratuita, se debe pagar por el uso de Machine Learning.
\$9,99 dólares mensuales más \$1 dólar por hora de procesamiento [22].	\$0,1 por cada 1000 predicciones en lote (redondeando hacia arriba) [23].	Su precio depende de las características de la Instancia [24].
Ofrece múltiples modelos de Machine Learning a disposición del usuario, aunque éste es el responsable de seleccionar el modelo adecuado para cada aplicación.	Ofrece un único tipo de modelo de Machine Learning para cada tipo de variable objetivo, sin embargo es la misma plataforma que lo selecciona de manera automática.	No ofrece modelos, aunque a cambio le da al usuario la opción de crear sus propios algoritmos usando Tensorflow.
Se usa programación en bloques para la creación de los proyectos.	Sigue una secuencia lógica en la que Amazon asiste en todo momento al usuario.	Se necesita programar de manera local utilizando Tensorflow para la creación de un modelo.
No necesita de otras herramientas para completar sus servicios de Machine Learning.	Necesita de otras aplicaciones (almacenamiento) para poder completar sus servicios de Machine Learning.	Necesita de otras aplicaciones (almacenamiento y trabajo local) para poder completar sus servicios de Machine Learning.
No se necesita de conocimientos previos para concretar un ejemplo.	No se necesita de conocimientos previos para concretar un ejemplo.	Se necesita saber Linux para poder concretar un ejemplo.
Se necesita saber JSON para editar las particiones del entrenamiento del modelo.	Se existe un bloque para crear particiones de entrenamiento y prueba.	Se necesita saber Python para programar las particiones de entrenamiento y prueba.
Su espacio de trabajo consta de un lienzo virtual en donde se van ubicando y conectando los bloques.	Su espacio de trabajo consta de distintas interfaces gráficas (de una misma página web) en donde cada una solicita al usuario cierta información específica.	Su espacio de trabajo consta de operar una máquina virtual a través de comandos de Linux junto con algunas interfaces gráficas (de la misma página web) que solicitan información específica.

A modo de resumen de la Tabla 5-5, se presentan algunas características comparativas referentes a las plataformas de Cloud Computing estudiadas.

- **Microsoft Azure Studio:**

- Posee servicios básicos de Machine Learning en su capa gratuita (con restricciones).
- Se necesita de una sola herramienta para generar aplicaciones de Machine Learning.
- Sencillo de utilizar gracias a que se programa de manera gráfica (uniendo bloques).
- Ofrece múltiples modelos de Machine Learning para probar diferentes alternativas.
- Ofrece bloques programables en Python y R para programar bloques personalizados.

- **Amazon Web Services:**

- Sus servicios son de pago, se paga por lo que se usa.
- Se necesita de un conjunto de herramientas que interactúan entre sí para generar aplicaciones.
- Sencillo de utilizar, gracias a que cuenta con un asistente guía al usuario en cada paso.
- Ofrece un modelo de Machine Learning según el tipo de la variable objetivo.
- Solo permite editar metadatos y ajustes del modelo ya impuesto por la plataforma.

- **Google Cloud Platform:**

- Sus servicios son de pago, se paga por lo que se usa.
- Se necesita de un conjunto de herramientas que interactúan entre sí para generar aplicaciones.
- Complejo de utilizar, se debe saber Linux para operar la consola virtual.
- No tiene modelos de Machine Learning, se deben crear localmente usando Tensorflow.
- Es posible personalizar tanto el modelo como sus metadatos.

Discusión y Conclusiones

Se comprueban los valiosos beneficios de la utilización de Cloud Computing, ya que se pudo procesar en un instante (alrededor de 1 minuto), una base de datos que ocasionaba problemas trabajarla (limitaciones por la cantidad de columnas máximas de Excel, junto con demoras e interrupciones al momento de extraer información de la base de datos) en un entorno local, dichos inconvenientes juegan un papel desfavorable (sobre todo para la investigación académica) para el usuario, ya que afecta fuertemente en los tiempos de espera invertidos. Otro punto a favor del Cloud Computing es la facilidad de acceso que se tiene, es posible acceder a los proyectos por cualquier dispositivo que se conecte internet (computadores, celulares, tablets, etc.)

Aunque los proveedores de los servicios de Cloud Computing ofrezcan una amplia variedad de aplicaciones (SaaS) que ayuden a resolver problemas genéricos que tengan los usuarios, no siempre se pueden cubrir las necesidades personalizadas de cada uno de cliente, es por ello que haber habilitado una plataforma virtual (PaaS) con la que los mismos usuarios pudieran crear sus propias aplicaciones fue una decisión completamente asertiva, ya que esto realimenta y potencia enormemente el uso de la nube y justifica la importancia que tienen hoy en día.

Al trabajar con servicios de Cloud Computing que disponen de Machine Learning, se comprueban nuevamente el gran provecho que se puede disfrutar de la combinación de estos productos, ya que no se necesita de ser un experto programador para realizar aplicaciones tales como lo es un clasificador de imágenes para un proyecto de fusión nuclear, ya que dichas plataformas virtuales ofrecen herramientas especializadas (algunas tienen modelos predefinidos), guías específicas, capacidad de trabajar en cualquier ubicación geográfica, velocidad de procesamiento superior al de computador promedio y una gran capacidad de almacenamiento cuyo respaldo está completamente garantizado. Por lo mencionado anteriormente, se espera que la industria de Cloud Computing siga creciendo, lo que la convertiría en una buena área de estudio para la ingeniería.

Analizando y comparando a los proveedores del servicio de Cloud Computing (Microsoft, Amazon y Google) desde el punto de vista académico, se llega a la conclusión de que a mejor plataforma (entiéndase que solo para fines de estudio) es Microsoft Azure Studio, ya que es una plataforma relativamente sencilla de utilizar (no se requiere de otros conocimientos previos), es

posible realizar proyectos en un tiempo relativamente corto, posee tutoriales y documentación que especifican de manera clara su uso, ofrece cierta variedad de modelos de Machine Learning de los cuales sus parámetros son completamente ajustables (tanto de manera manual como automática) y sus servicios de Machine Learning se encuentran en la capa gratuita, dando la oportunidad a los estudiantes de repetir sus experimentos las veces que estime necesario sin la necesidad de preocuparse ni restringirse por algún tipo de costo.

Al observar los resultados obtenidos de los experimentos realizados sobre el clasificador de imágenes Thomson, se puede notar que los factores de mayor influencia sobre el porcentaje de acierto para una aplicación con Machine Learning son el modelo a utilizar y la técnica de extracción de características. Esto se debe al esencial aporte que contribuyen estos elementos al sistema: la extracción de características es la forma de procesar la “materia prima” (información total disponible), resultando en la información de entrada que recibe la máquina; el modelo de Machine Learning es “la forma de pensar” que tendrá la máquina al momento de procesar la información.

Bibliografía

- [1] Salesforce, «Salesforce,» [En línea]. Available: <https://www.salesforce.com/mx/cloud-computing/>. [Último acceso: 05 Diciembre 2017].
- [2] R. X. C. Quispe, «Cloud Computing para Aplicaciones Logísticas,» *evistas Bolivianas*, nº 7, 2012.
- [3] M. A. Paz Pellat, «slideshare,» 7 Septiembre 2010. [En línea]. Available: <https://es.slideshare.net/politicadospuntocero/cmputo-en-la-nube-marco-antonio-paz-pellat>. [Último acceso: 24 Abril 2018].
- [4] J. E. V. Reyna, «Cloud Computing,» de *Congreso internacional de Innovación Educativa*, 2009.
- [5] R. Barranco, «IBM,» 18 Junio 2012. [En línea]. Available: <https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/index.html>. [Último acceso: 2017 Diciembre 2017].
- [6] P. Recuero, «Blog Think Big,» 29 Octubre 2017. [En línea]. Available: <https://blogthinkbig.com/artificial-intelligence-machine-learning-y-deep-learning-conoces-las-diferencias>. [Último acceso: 14 Enero 2018].
- [7] F. S. Caparrini, «Dpto. de Ciencias de la Computacion e Inteligencia Artificial de la Universidad de Sevilla,» 23 Septiembre 2017. [En línea]. Available: <http://www.cs.us.es/~fsancho/?e=75>. [Último acceso: 14 Enero 2018].
- [8] Microsoft, «Microsoft,» 10 Agosto 2016. [En línea]. Available: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-forest>. [Último acceso: 28 noviembre 2017].

-
- [9] M. Bihis, «A generalized Flow for Multi-class and Binary Classification Task: An Azure ML Approach,» de *Big Data (Big Data), 2015 IEEE International Conference on*, Santa Clara, CA, USA, 2015.
- [10] H. Hong, W. Tong, R. Perkins, H. Fang, Q. Xie y L. Shi, «Multiclass Decision Forest—A Novel Pattern Recognition Method for Multiclass Classification in Microarray Data Analysis,» *DNA and Cell Biology*, vol. 23, n° 10, p. 685–694, 2004.
- [11] Gunarathne, Perera y Kahandawaarachchi, «Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease,» de *International Conference on Bioinformatics and Bioengineering*, Washington DC, 2017.
- [12] M. Á. Dueñas Rodríguez, «Modelos de Respuesta Discreta en R y Aplicaciones con Datos Reales,» Granada.
- [13] KDnugget, «KDnugget,» 2016. [En línea]. Available: <https://www.kdnuggets.com/2016/08/beginners-guide-neural-networks-r.html>. [Último acceso: 24 Abril 2018].
- [14] J. Cancela, «Javier Cancela,» 19 Marzo 2017. [En línea]. Available: <https://www.javiercancela.com/2017/03/19/python-machine-learning-ii/>. [Último acceso: 24 Abril 2018].
- [15] M. G. Pose, «Introducción a las Redes Neuronales Artificiales,» Coruña, 2009.
- [16] J. Riquelme, R. Ruiz y K. Gilbert, «Minería de Datos: Conceptos y Tendencias,» *Revista Iberoamericana de Inteligencia Artificial*, vol. 10, n° 29, pp. 11-18, 2006.
- [17] L. C. M. Félix, «UOC (Universitat Oberta de Catalunya),» Noviembre 2002. [En línea]. Available: <http://www.uoc.edu/web/esp/art/uoc/molina1102/molina1102.html>. [Último acceso: 13 Enero 2018].
- [18] G. Farias, S. Dormido-Canto, J. Vega, I. Martínez, L. Alfaro y F. Martínez, «Fusion Engineering and Desing,» *Elsevier*, n° 123, pp. 759-763, 2017.
- [19] G. Farias, S. Dormido-Canto, J. Vega, I. Martínez, L. Alfaro y F. Martínez, «Fusion Engineering and Desing,» *Elsevier*, vol. 123, pp. 759-763, 2017.
- [20] G. Farías, S. Dormido-Canto, J. Vega, G. Rattá, H. Vargas, G. Hermosilla, L. Alfaro y A. Valencia, «Automatic feature extraction in large fusion databases by using deep learning approach,» *Elsevier*, vol. 112, pp. 979-983, 2016.

- [21] R. Barga, V. Fontama y W.-H. Tox, Predictive Analytics with Microsoft Azure Machine Learning, Second Edition, Apress, 2015.
- [22] Microsoft, «Microsoft Azure,» 2017. [En línea]. Available: <https://azure.microsoft.com/en-us/pricing/details/machine-learning-studio/>. [Último acceso: 14 Diciembre 2017].
- [23] Amazon, «Amazon Web Services,» 2017. [En línea]. Available: <https://aws.amazon.com/es/aml/pricing/>. [Último acceso: 14 Diciembre 2017].
- [24] Google, «Google Cloud Platform,» 11 Diciembre 2017. [En línea]. Available: <https://cloud.google.com/compute/pricing?hl=es>. [Último acceso: 14 Diciembre 2017].

A Apéndice

A continuación, se presentan los códigos de programación utilizados en este proyecto.

- Script en Python usado para la creación de los 1000 datos aleatorios.

```
import pandas as pd
import numpy as np

dataSize = 1000    #cantidad de datos aleatorios
maxNumber = 100

#se generan valores aleatorios entre 0 y 6
X1 = np.random.randint(maxNumber + 1, size = dataSize)*6.0/100.0
x2 = np.random.randint(maxNumber + 1, size = dataSize)*6.0/100.0

df = pd.DataFrame(dict(X1=X1, x2=x2)) #se combinan los valores
creados de tal manera de generar coordenadas cartesianas

pd.DataFrame.to_csv(df, 'datos sin etiqueta POTATO.csv', index=False)
#se duardan los valores creados como un archivo ".csv"
```

- Script en Python usado para la creación del Dataset del clasificador de imágenes Thomson.

```
import pandas as pd
import numpy as np

pixeles = pd.read_csv('C:/Users/YOYO/Desktop/Thomson
AWS/Dataset/BaseDatosSC.csv')
categoria = pd.read_csv('C:/Users/YOYO/Desktop/Thomson
AWS/Dataset/DataSetClases.csv')

finalDataFrame = np.matrix('')
agruX = 15    #cantidad de pixeles agrupados en el eje Y
para la reduccion
agruY = 16    #cantidad de pixeles agrupados en el eje X
para la reduccion
auxX = 576    #cantidad de pixeles en el eje X que posee una
imagen
```

```

auxY = 385 #cantidad de pixeles en el eje Y que posee una
imagen
matfinal = np.matrix('')
DatosBrutos = np.array(pixeles) #se asigna la base de datos a
una variable
(filas,columna) = DatosBrutos.shape #se detectan las
dimensiones de la matriz de la base de datos
DatosBrutos.T
for i in range(columna): #se realiza el siguiente
proceso para cada imagen
    aux = DatosBrutos[:,i]
    ordeM = np.reshape(aux,(auxY,auxX),order='F') #cada vector
correspondiente a una imagen, se transforma en una
    ordeM = np.rot90(ordeM)

    newX = auxX//agruX #valor que indica de manera entera,
cuantas agrupaciones de pixeles caben en el eje X
    newY = auxY//agruY #valor que indica de manera entera,
cuantas agrupaciones de pixeles caben en el eje Y

    neoX = newX*agruX #dimension X que tendria la nueva matriz
de parametros
    neoY = newY*agruY #dimension Y que tendria la nueva matriz
de parametros

    mataux = np.zeros((neoX,neoY)) #se genera una matriz de
dimensiones especificas para guardar la informacion a trabajar
    for x in range(neoX):
        for y in range(neoY):
            mataux[x,y] = ordeM[x,y] #se llena la matriz

    matacom = mataux.reshape(newX,agruX,newY,agruY) #se reduce la
matriz comprimiendo los datos
    matreduct = matacom.mean(axis=(1,3))

    vectaux = matreduct.ravel() #se transforma la matriz reducida
en un vector
    vectaux = np.matrix(vectaux) #se cambia el formato de la
variable trabajada

#se agregan los vectores como elementos para la formacion de una
matriz
    if matfinal.size == 0:
        matfinal = np.append(matfinal,vectaux,axis=1)

    else:
        matfinal = np.append(matfinal,vectaux,axis=0)

clasesMatrix = np.matrix(categoria.values)
#Juntamos los datos y las clases
finalMatrix = np.append(matfinal, clasesMatrix, axis = 1)
#Se crea la lista de etiquetas (nombres de las columnas)
col = []

```

```

for i in range(finalMatrix[0,:].size - 1): #Número de datos(pixeles
reducidos)
    col.append('pixel '+str(i))
col.append('class')
#Por último, se crea el DataFrame
finalDataFrame = pd.DataFrame(finalMatrix, columns = col)

pd.DataFrame.to_csv(finalDataFrame,'DataSet      Thomson      8x7.csv',
index=False)

```

- Código de Python usado para la decodificación de los resultados entregados por la plataforma para la presentación de los datos finales de los datos finales.

```

##### Decodificacion de los resultados
#####

data =
pd.read_csv('C:/Users/YOYO/Desktop/NewNeoAdaBoost/Resultados/Resultados
s Adaboost.csv')
deco = data.columns
predict = {}
for i, d in enumerate(data.values):
    predict[i] = deco[np.argmax(d)]

##### Preparacion final de los datos
#####
coordinates = pd.read_csv('C:/Users/YOYO/Desktop/AWS/Datos/Datos sin
Etiqueta.csv') #se leen los datos originales

predictions = pd.DataFrame.from_dict(predict, orient='index',
dtype=None) #se transforma la prediccion de los datos a un dataframe
predictions.columns = ['clase'] #le cambia el nombre de la columna

results = pd.concat([coordinates, predictions], axis=1) #se etiquetan
las coordenadas iniciales segun lo indicado por el modelo

##### Ploteando los resultados decodificados
#####
coordinates = pd.read_csv('C:/Users/YOYO/Desktop/AWS/Datos/Datos sin
Etiqueta.csv') #se leen los datos originales

predictions = pd.DataFrame.from_dict(predict, orient='index',
dtype=None) #se transforma la prediccion de los datos a un dataframe
predictions.columns = ['clase'] #le cambia el nombre de la columna

results = pd.concat([coordinates, predictions], axis=1) #se etiquetan
las coordenadas iniciales segun lo indicado por el modelo

```

- Código de Python usado para mostrar de manera gráfica la base de datos usados en el etiquetador de clases.

```
dataset =
pd.read_csv('C:/Users/YOYO/Desktop/AWS/Datos/DatosAdaBoost.csv')

groups0 = dataset.groupby(by='clase')
markers = ['o', 'x']

# Plot
fig, ax = plt.subplots()
ax.margins(0.1) # grafica un 10% mas de lo necesaiio, para que no se
vea tan justo
for (name, group), marker in zip(groups0, cycle(markers)):
    ax.plot(group.x1, group.x2, marker=marker, linestyle='', ms=4,
label=name)
ax.legend(numpoints=1, loc='best')

fig.savefig('figure2.png')
plt.title('Base de datos original')
plt.show()
```

- Código en Python utilizado para la creación del modelo Red Neuronal usando Tensorflow.

```
from tensorflow.contrib.keras.api.keras.layers import Input, Dense
from tensorflow.contrib.keras.api.keras.models import Model
import pandas as pd
import numpy as np

df = pd.read_csv('C:/Users/yoyo_/Desktop/Tensorflow/datos.csv')

labels = df['clase']
encoding_label = {}
for ii, clase in enumerate(list(set(labels))):
    encoding_label[clase] = ii*2-1
for ii, label in enumerate(labels):
    labels[ii] = encoding_label[label]
X = df[['X1', 'x2']]

# modelo
inputs = Input([2])
x = Dense(64, activation='tanh')(inputs)
x = Dense(64, activation='tanh')(x)
out = Dense(1, activation='tanh')(x)

model = Model(inputs, out, 'mnist')
model.compile('adam', 'mean_absolute_error', metrics=['acc'])
model.summary()

# train
```

```
model.fit(x=X.values, y=labels, epochs=1000, verbose=2)
# test
x_test = np.random.rand(1000,2) * 6

predict = model.predict(x_test)

df = pd.DataFrame(np.concatenate([x_test, np.reshape(predict,[-1,1])],
axis=1), columns=['x', 'y', 'predict'])
y = df[df['predict'] > 0]
x = df[df['predict'] <= 0]
x = x[['x', 'y']]
y = y[['x', 'y']]
from matplotlib import pyplot as plt
plt.figure(figsize=[10,10])
plt.scatter(x['x'], x['y'], marker='X', color='orange')
plt.scatter(y['x'], y['y'], color='b')
```